Pervasive Handheld Computing Systems

Dissertation submitted to the National University of Ireland, Cork
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy of Computer Science

October 2005

Timothy O' Sullivan
Department of Computer Science
University College Cork
Ireland

ABSTRACT

The technological role of handheld devices is fundamentally changing. Portable computers were traditionally application specific. They were designed and optimised to deliver a specific task. However, it is now commonly acknowledged that future handheld devices need to be multi-functional and need to be capable of executing a range of high-performance applications. This thesis has coined the term *pervasive handheld computing systems* to refer to this type of mobile device.

Portable computers are faced with a number of constraints in trying to meet these objectives. They are physically constrained by their size, their computational power, their memory resources, their power usage, and their networking ability. These constraints challenge *pervasive handheld computing systems* in achieving their multi-functional and high-performance requirements.

This thesis proposes a two-pronged methodology to enable *pervasive handheld computing systems* meet their future objectives. The methodology is a fusion of two independent and yet complementary concepts. The first step utilises reconfigurable technology to enhance the physical hardware resources within the environment of a handheld device. This approach recognises that reconfigurable computing has the potential to dynamically increase the system functionality and versatility of a handheld device without major loss in performance. The second step of the methodology incorporates agent-based middleware protocols to support handheld devices to effectively manage and utilise these reconfigurable hardware resources within their environment.

The thesis asserts the combined characteristics of reconfigurable computing and agent technology can meet the objectives of *pervasive handheld computing systems*.

DECLARATION

This dissertation is submitted to University College Cork, in accordance with the requirements for the degree of Doctor of Philosophy in the Faculty of Science. The research and thesis presented in this dissertation are entirely my own work and have not been submitted to any other university or higher education institution, or for any other academic award in this university. Where use has been made of other people's work, it has been fully acknowledged and referenced.

Excerpts of this thesis have been published in journals, conference and workshop papers, namely:

[O' Sullivan, 6a], [O' Sullivan, 6b], [O' Sullivan, 6c], [O' Sullivan, 6d], [O' Sullivan, 6e], [O' Sullivan, 5a], [O' Sullivan, 5b], [O' Sullivan, 5c], [O' Sullivan, 5d], [O' Sullivan, 5e], [O' Sullivan, 4a], [O' Sullivan, 4b], and [O' Sullivan, 4c].

Timothy O' Sullivan
October 2005.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor Dr. Richard Studdert for providing me with the opportunity to undertake this work. His continual motivation, guidance, and friendship throughout my studies in UCC have been tremendous. I will be forever grateful for his encouragement and kindness.

I would also like to acknowledge all the support and time which Dr. John Herbert provided during the course of my studies. His insight and knowledge was greatly appreciated.

I am also extremely appreciative of the financial support provided to me through the Boole Centre for Research in Informatics.

Finally, I would like to thank my family for their help and encouragement. Special thanks go to Alice and Mum whose friendship and love made this thesis possible.

DEDICATION

To my Father and Mother for teaching me the important things in life

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACL	Agent Communication Language
AES	Advanced Encryption Standard
AOAD	Agent-Oriented Analysis and Design
AOSE	Agent-Oriented Software Engineering
API	Application Program Interface
ASIC	Application Specific Integrated Circuit
AUML	Agent Unified Modelling Language
BDI	Belief, Desire, Intention
CAMMD	Context-Aware Mobile Medical Devices
CLB	Configurable Logic Block
CLIPS	C Language Integrated Production System
DES	Data Encryption Standard
DICOM	Digital Imaging and Communications in Medicine
DLL	Dynamic Link Libraries
DRIP	Dynamically Reconfigurable Image Processor
EDIF	Electronic Database Interchange Format
EJB	Enterprise Java Bean
FIPA	Foundation for Intelligent Physical Agents
FIPA-ACL	FIPA - Agent Communication Language
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HDL	Hardware Description Language
IDEA	International Data Encryption Algorithm

IOB	Input-Output Block
IRL	Internet Reconfigurable Logic
JADE	Java Agent Development Environment
JADE-LEAP	JADE-Lightweight Extensible Agent Platform
JADEX	Jade eXtension
JESS	Java Expert System Shell
JNI	Java Native Interface
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
KQML	Knowledge Query and Manipulation Language
MAC	Media Access Control
MASB	Multi-Agent Scenario-Based
MASE	Multiagent Systems Engineering
Mbps	Megabits per second
Ms	Millisecond
OMT	Object Modelling Technique
PAVE	PLD API VxWorks Embedded
PEP	Packet Exchange Platform
QoS	Quality of Service
RARE	Remote Adaptive computing Resource Environment
RMI	Remote Method Invocation
RSA	Rivest, Shamir, and Adelman
RTL	Register Transfer Level
SDR	Software Defined Radio
SMS	Short Messaging System

TCP / IP	Transmission Control Protocol / Internet Protocol
UML	Unified Modelling Language
USB	Universal Serial Bus
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
WAP	Wireless Application Protocol
Wi-Fi	Wireless Fidelity
XHWIF	Xilinx HardWare InterFace
XML	eXtensible Markup Language
XNF	Xilinx Native Files
XSL	Extensible Stylesheet Language

CHAPTER 1

Introduction

1.1 Introduction

The technological role of handheld devices is fundamentally changing. Portable computers were traditionally application specific. They were designed and optimised to deliver a specific task. However, it is now commonly acknowledged that future handheld devices need to be multi-functional and need to be capable of executing a range of high-performance applications [Fleischmann, 99].

This technological shift in handheld device usage is occurring within a number of computing environments. Forthcoming generations of mobile devices within the telecommunications market are expected to execute multiple tasks concurrently [Mignolet, 02]. These tasks will include audio streaming, video encoding/decoding and graphical processing.

The medical community constitutes another computing environment which is experimenting with incorporating handheld device technology into healthcare systems. This specialised field has recognised that greater levels of patient care can be achieved through mobile computing [Kroll, 02]. Enabling nomadic healthcare professionals with access to efficient medical tools at locations of their choice enhances their productivity levels and improves the accuracy of their patient diagnosis [Ancona, 01]. Essentially, multi-functional and high-performance portable computers allow the medical community exploit the capabilities of ubiquitous computing.

This ubiquitous vision of “everywhere, anytime” computing is a primary driving force behind the changing technological role of handheld devices [Weiser, 93].

The primary concept within ubiquitous computing[1] research is that the computer *disappears* and computing services are made available to users throughout their physical environment [Weiser, 91]. Handheld devices are a key component in realising these ubiquitous computing environments.

Whereas today's notebook computers and personal digital assistants are self-contained, tomorrow's networked mobile computers will act as access points to greater computing infrastructures [Smit, 02]. These mobile computers will also be capable of supporting a wide range of multimedia-based functionality [Smit, 01]. This thesis has coined the term *pervasive handheld computing systems* to refer to this type of mobile device.

The technological challenges raised by the vision of ubiquitous computing environments create serious performance and versatility issues for *pervasive handheld computing systems*. These portable computers will be multi-functional and capable of executing a broad range of compute intensive applications. These handheld devices will employ a sophisticated middleware framework to facilitate effective inter-communication and management structures.

Handheld devices are faced with a number of constraints in trying to meet these objectives. They are physically constrained by their size, their computational power, their memory resources, their power usage, and their networking ability. These constraints challenge *pervasive handheld computing systems* in achieving their multi-functional, high-performance requirements and jeopardise their role within the ubiquitous computing vision.

Clearly, this changing technological role of handheld devices means the scope of functionality and versatility required by *pervasive handheld computing systems* is expanding. The consequence of this progression is the need for more proficient design and deployment methodologies to help address the technological challenges ahead.

[1] Also known as Pervasive or Ambient Computing

The methodologies currently employed by researchers can generally be categorised into two distinct groupings:

- Enhancement of Hardware Resources within a Handheld Device Environment

  This approach seeks to improve the performance and versatility of *pervasive handheld computing systems* by seeking to enhance the quality of hardware resources within the environment of a mobile device. These resources are either available as an integral component of a handheld device or are accessible as a networked resource within a distributed environment. Examples of research attempting to enhance integral hardware components within a portable computer can be seen in [Mignolet, 03], [Smit, 01], and [Guccione, 02]. Examples of research focusing on enhancing hardware resources within a distributed network are [Doss, 99], [Smith King, 01], and [Gaj, 03].

- Enhancement of Middleware Solutions for Handheld Computing

  This approach focuses on *pervasive handheld computing systems* resolving ubiquitous computing challenges by developing sophisticated middleware infrastructures. These middleware frameworks are designed to operate within mobile computing environments. They seek to enhance performance and versatility of portable computers by providing them with adept inter-communication capabilities. These middleware frameworks allow handheld devices to engage in efficient exchange of information and data with other distributed entities within a network environment. This facilitates effective management of network resources.

  A sophisticated middleware is also perceived as a crucial aspect in constructing contextually-aware handheld devices. Portable computers with contextual capabilities are able to monitor their environment and are able to adapt to changes within it. This allows for comprehensive management and utilisation of hardware resources within handheld device environments. Examples of research focusing on developing middleware solutions within handheld computing environments can be found in [Chalmers, 98], [Edmonds, 01], and [Capra, 02].

This thesis stresses the need to combine both methodologies to create a uniform approach. The thesis argues that the challenges posed by the ubiquitous computing vision would be better resolved with this combined strategy.

This two-pronged approach places an equal emphasis upon the need for high-performance hardware resources within handheld device environments and upon a sophisticated middleware to enable effective management of these aforementioned resources.

The thesis argues that this methodology constitutes a more coherent approach to meeting the objectives of *pervasive handheld computing systems*.

1.2 Methodology Overview

The proposed methodology is a fusion of two independent and yet complementary concepts. The first step focuses upon utilising reconfigurable technology to enhance the physical hardware resources within the environment of a handheld device. The key feature of reconfigurable computing is its ability to perform computations in hardware to increase performance, whilst retaining much of the flexibility of a software solution [Compton, 02].

The methodology integrates reconfigurable hardware into the environment of a handheld device i.e. both into the physical device and into surrounding adaptive servers. This approach recognises that reconfigurable computing has the potential to dynamically increase the system functionality and versatility of a handheld device without major loss in performance.

The second step of the proposed methodology seeks to incorporate sophisticated middleware protocols to support handheld devices to effectively manage and utilise reconfigurable hardware resources within their environment. Agent technology is the mobile computing middleware employed [Wooldridge, 97].

Agents operating within physical components in a distributed network can be engineered with adept decision-making capability. This characteristic facilitates intelligent and proactive utilisation of reconfigurable hardware-based resources.

Agents are particularly well-suited to wireless networks as they are efficient in their use of bandwidth and they can deal with intermittent network connections. Additionally, an agent can effectively represent, communicate and work towards a user's preferences and interests. Crucially, the agent concept provides a high-level of abstraction that is favourable towards the development of complex distributed heterogeneous systems.

The thesis asserts the combined characteristics of reconfigurable computing and agent technology can meet the goals of *pervasive handheld computing systems*. Their combination represents a uniform and coherent approach to addressing the fundamental challenges of ubiquitous computing.

The proposed methodology incorporates reconfigurable hardware as an integral component within a handheld device and as an external resource within its distributed environment. The methodology employs an agent-based middleware to empower handheld devices to effectively utilise these reconfigurable resources.

1.3 Contributions

This dissertation makes a number of research contributions within the field of *pervasive handheld computing systems*. The major contribution of this work is the development of agent-based middleware protocols which support handheld devices in exploiting reconfigurable resources within their environment.

These middleware protocols are categorised into two groupings according to reconfigurable hardware placement, namely:

- Employing reconfigurable hardware as a networked resource
- Employing reconfigurable hardware as an integral handheld device component

This work also contributes to the field of agent research for *pervasive handheld computing systems* by presenting a mixed agent-object design methodology.

Additionally, the thesis establishes two middleware frameworks targeted towards agent-based mobile computing environments. These frameworks promote the learning, adaptive, and collaborative nature of agent-based systems. The middleware frameworks also highlight the valuable contribution agent technology can make within *pervasive handheld computing systems*.

1.4 Dissertation Structure

The thesis is structured as follows:

- Chapter Two reviews relevant research within the context of this thesis. This includes a detailed examination of the concept of personal mobile computing. The fundamental challenges and identifying characteristics of this research paradigm are outlined.
  An overview of the *pervasive handheld axis* conceptual model is presented. This model conveys those elements considered crucial in the construction of *pervasive handheld computing systems*. An examination of related research which focuses upon enhancing the *hardware resources* and *middleware technology* elements of the *Pervasive Handheld Axis* is given.
  As the prototype systems constructed within the context of this thesis are targeted towards medical environments, a detailed assessment of the role of mobile device technology within telemedicine is also presented.

- Chapter Three explores the field of reconfigurable computing within the context of mobile computing environments. Reconfigurable technology is the enabling hardware proposed within *pervasive handheld computing systems*.
  The chapter consists of a short history of the origins of reconfigurable computing as well a detailed overview of its technological components. The software tools required to effectively construct and deploy reconfigurable hardware-based solutions are investigated.

An appraisal of the performance capabilities of reconfigurable hardware in comparison with software-based solutions with special emphasis placed upon applications within the medical field is presented.

The chapter concludes with an examination of related research which has utilised reconfigurable logic either as an integral handheld device component or as a resource within a networked environment.

- Chapter Four is concerned with the concept of agent-based computing. Agents are the proposed middleware solution within *pervasive handheld computing systems*. This chapter clearly defines the agent concept and identifies the unique characteristics associated with agent-based software engineering.

  An overview of agent-based development systems is presented. This includes a detailed examination of the agent frameworks employed within the medical-based prototype systems within this thesis. Agent-oriented analysis and design techniques are also explored.

  The chapter also contains an appraisal of the important differences between the concepts of agent-oriented and object-oriented programming. Encapsulating these differences is the proposed mixed agent-object design technique which was conceived to improve performance of agent-based systems [O' Sullivan, 06e].

  Additionally, an examination of related work primarily exploring agent-based middleware solutions deployed within mobile computing environments is outlined.

- Chapter Five focuses on the importance of facilitating *pervasive handheld computing systems* with context-aware capabilities within medical environments. All prototype systems developed and presented within this thesis are constructed to operate within the medical field. Each experimental prototype comprises of a handheld medical device executing an agent-based middleware protocol to enable utilisation of reconfigurable resources within its environment.

  A number of these prototype systems sense and interpret their contextual environment to facilitate the deployment framework with enhanced decision-making ability. Thus, the chapter initially explores the concept of context-aware computing.

The role of context-awareness within handheld medical device environments is highlighted through a description of the developed CAMMD[2] framework [O' Sullivan, 06d]. This deployment scenario outlines the capability of a context-aware, agent-based framework to enhance the usability and portability of mobile medical devices. The proposed methodology is also shown to have the capability to address both storage and network bandwidth constraints.

- Chapter Six is concerned with investigating the potential of agent technology to exploit reconfigurable resources for *pervasive handheld computing systems*. The reconfigurable hardware is integrated into the environment of the portable computer, i.e. it is placed both into the physical device and into surrounding adaptive servers. A number of middleware protocols are proposed:

    - An agent-based negotiation and bidding protocol enabling handheld devices gain efficient access to networked reconfigurable resources [O' Sullivan, 05c], [O' Sullivan, 05d].
    - A context-based negotiation protocol enabling utilisation of networked reconfigurable resources by handheld devices within contextual environments [O' Sullivan, 05a].
    - An application solutions retrieval protocol for handheld devices [O' Sullivan, 04c].
    - The establishment of a push-based configuration management strategy for the coherent distribution of reconfigurable-hardware based application solutions to handheld devices [O' Sullivan, 04a].
    - The introduction of context-aware hardware reconfiguration protocol for handheld devices. This enhances utilisation of reconfigurable resources contained within a portable computer by

[2] Context-Aware Mobile Medical Devices

initiating a configuration process according to the application requirements of a user [O' Sullivan, 06b].

These protocols provide a firm basis to explore the potential of the proposed methodology in meeting the challenges of *pervasive handheld computing systems*.

- Chapter Seven presents a negotiation protocol that incorporates learning and adaptive characteristics into an agent-based framework [O' Sullivan, 06a]. The integration of these agent attributes into a development system improves utilisation of networked reconfigurable resources by handheld medical devices.

  The chapter also details a collaborative learning strategy that enhances the knowledge base of negotiating agents [O' Sullivan, 06c]. This strategy employs the physical trait of agent mobility to pool knowledge resources and to effectively disseminate accumulated experiences.

- Chapter Eight presents an overall evaluation of the presented work summarising the contributions made within this thesis. A detailed overview of future work within the field of *pervasive handheld computing systems* is also outlined.

CHAPTER 2

Background and Related Work

2.1 Introduction

The field of personal mobile computing will play a significant role in driving technology in the next decade [Smit, 02]. Mobile devices are becoming more powerful, ubiquitously available, and network-connected [Mahmoud, 02]. These technological advancements are leading to an increasing number of portable computers within physical environments.

This proliferation of mobile computing devices will help realise the ubiquitous computing vision of people and environments augmented with computational resources [Abowd, 02]. The realisation of this vision is reliant upon the computational resources having the capability to provide information and services when and where desired.

The demands placed upon computational resources within ubiquitous environments are fundamentally changing the technological role of handheld devices. The scope of functionality and versatility required by *pervasive handheld computing systems* is expanding. This expansion is however impeded by the constraints associated with portable devices.

These constraints are unique to the field of personal mobile computing and are as follows [Satyanarayanan, 96]:

- Resource Poor Mobile Elements
  Portable device constraints of weight, power, size and ergonomics negatively affect computational resources such as processor speed, memory size, and disk capacity [Gaddah, 03]. Whilst mobile elements will improve in absolute ability, they will always be resource-poor relative to static elements.

- Highly Variable Network Connectivity

  Handheld devices suffer temporary and unannounced loss of network connectivity through physical movement [Mascolo, 02], [Mascolo, 04]. The two major factors causing network instability are the nature of the wireless medium and the mobility of devices [Sterbenz, 02]. Interference can occur within wireless communication channels because of atmospheric disturbances such as electromagnetic storms or environmental factors such as rain [Kurkovsky, 04].

  Essentially, low data bandwidths and intermittent network connections are environmental constraints synonymous with portable computers. These delimiting factors affect device usage and can lead to an unsatisfactory user experience.

- Limited Power Resources

  Next generation applications deployed on wirelessly networked embedded devices will operate with extremely low power budgets [Verbauwhede, 02]. Portable computers will need to deliver functionality with optimal energy efficiency [Smit, 99].

These fundamental constraints present considerable limitations within the field of personal mobile computing. They challenge the multi-functional and high-performance requirements of *pervasive handheld computing systems*. The research methodologies currently employed by researchers to meet the versatility and performance objectives of *pervasive handheld computing systems* can generally be categorised into two distinct groupings.

The first approach has focused on enhancing hardware resources within the environment of a handheld device. The second approach has prioritised the importance of developing sophisticated middleware solutions for handheld computing environments. These middleware solutions enhance handheld device performance by introducing protocols which improve utilisation of hardware resources. Both of these methodologies have their own individual merits.

Previous research has witnessed only small degrees of emphasis placed upon the need to employ both approaches to help resolve the challenges of *pervasive handheld computing systems*. This thesis stresses the need to combine both methodologies to create a uniform approach. The thesis argues a two-pronged approach can better resolve the challenges posed by the ubiquitous computing vision.

2.2 Pervasive Handheld Axis

Three key components have been identified as essential elements in constructing *pervasive handheld computing systems*. These form an axis of inter-dependability known as the *Pervasive Handheld Axis* which is presented in Figure 2.1.

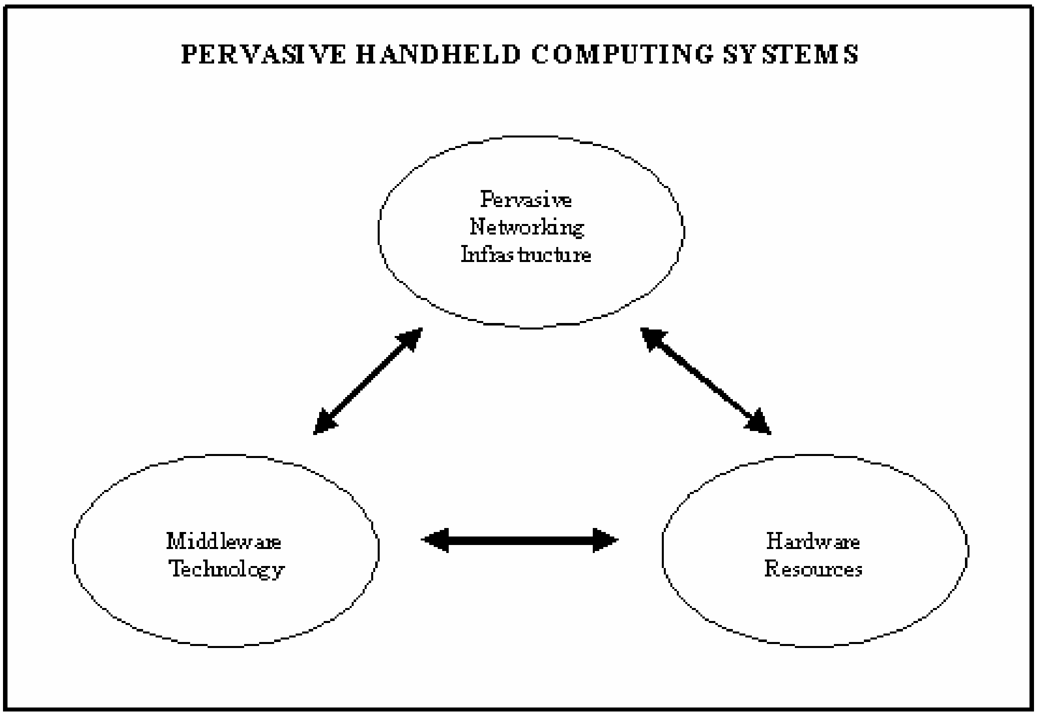

PERVASIVE HANDHELD COMPUTING SYSTEMS
Pervasive Networking Infrastructure
Middleware Technology
Hardware Resources

Figure 2.1: Pervasive Handheld Axis for Mobile Computing Systems

The two-pronged approach of the proposed methodology caters for the *middleware technology* and *hardware resources* elements of the *Pervasive Handheld Axis.* The methodology considers the *pervasive networking infrastructure* element to be a secondary factor as networking infrastructures are becoming increasingly omnipresent.

Within the context of this work, the *middleware technologies* element of the *Pervasive Handheld Axis* relates to handheld devices employing middleware protocols for utilisation of reconfigurable resources. These proposed protocols constitute the second part of our proposed methodology enabling *pervasive handheld computing systems* meet their future objectives.

A range of related research has focused on enhancing the *hardware resources* and *middleware technology* elements of the *Pervasive Handheld Axis*. These research efforts are outlined in Section 2.3 and 2.4 respectively.

2.3 Hardware Resources within Handheld Device Environments

Performance and versatility are two key issues gathering increasing attention within the handheld device community. Reconfigurable technology is considered by many as a viable solution to address these portable computer concerns. The reconfigurable computing concept is explored in-depth in Chapter 3. Specific examples of the performance capabilities of reconfigurable hardware are presented in Section 3.7.

The methodologies employed by mobile computing research groups utilising reconfigurable hardware can be categorised into two groupings, namely:

- Integral Component Placement
  This entails placing reconfigurable hardware within a portable device to operate as an integral component. Reconfigurable technology incorporated within a handheld device is beneficial as it establishes a flexible and dynamic system architecture which can help achieve the requirements of *pervasive handheld computing systems*.

- Networked Resource Placement
  This involves introducing reconfigurable hardware as an external resource within the networked environment of a handheld device. Reconfigurable hardware usage is requested by the portable computer. This enables a handheld device to offload compute-intensive computational tasks to neighbouring reconfigurable hardware-based servers.

2.3.1 Integral Component Placement

Multimedia-based applications executing on handheld computers require high performance capabilities to achieve efficient processing of compute-intensive applications [Singh, 94]. A reconfigurable computing platform within a networked and portable multimedia appliance can increase computational power and maintain device flexibility [Guccione, 02].

The Chameleon research project focuses on developing a heterogeneous reconfigurable mobile system [Smit, 01]. Their device architecture operates in cohesion with a QoS[3]-driven operating system. This facilitates selecting granularity of computation in accordance with the model of task to be performed i.e. the physical architecture is matched dynamically to each application.

An example application of this design methodology is fine-tuning the settings of a SDR[4] component within a handheld device [Smit, 02]. In contrast to ASIC[5] implementations, these radio modifications can be made at runtime according to the context of the external environment. This translates to reductions in the energy consumption of a mobile device.

Research within the Gecko project has produced a reconfigurable hardware-based video decoder application operating within a handheld device [Mignolet, 03]. The video decoder is a motion JPEG frame decoder. The methodology employed supports the relocation of the application from reconfigurable hardware to software and vice-versa. This approach allows the handheld operating system to spawn or relocate tasks and also enables QoS management.

Additional research utilising reconfigurable technology as an integral handheld device component has ranged from digital signal processing components for audio MP3 players [Scalera, 00] to video compression tools [Lee, 00] to enhanced teaching tools within university microcomputer courses [Gottlieb, 03].

[3] Quality of Service

[4] Software Defined Radio

[5] Application Specific Integrated Circuit

2.3.2 Networked Resource Placement

A range of research projects have constructed system architectures to facilitate utilisation of networked hardware resources by client machines. These research efforts have seldom focused upon mobile computing environments. Instead, the client machines have tended to be represented by workstations which are connected to networked adaptive servers over reliable wired networks [Smith King, 01], [Casselman, 02].

Clearly, research projects dealing with networked reconfigurable hardware rely upon a middleware framework. However, it has been found that middleware support is commonly treated as an aside within the overall objectives of these projects. It is generally merely utilised to establish a connection to facilitate simple inter-communication between clients and servers.

Essentially, middleware frameworks are under-exploited in terms of establishing protocols to facilitate optimal utilisation of networked hardware resources. This is especially prevalent for resource-constrained handheld devices seeking to offload compute-intensive tasks to neighbouring adaptive servers.

Research projects which have investigated middleware frameworks to support network resource placement are outlined within the middleware technology section of 2.4.2.

2.4 Middleware Technology

Pervious research incorporating reconfigurable hardware within a distributed environment as a networked resource has tended to employ simple forms of middleware infrastructure. Research efforts using reconfigurable technology as an integral handheld device component have also employed simplified middleware infrastructures.

These middleware solutions have tended to model traditional client-server frameworks. These infrastructures are inappropriate for the dynamic and sporadic nature of mobile computing environments.

Furthermore, there is a lack of research investigating middleware protocols that target effective exploitation of reconfigurable resources within handheld device environments. Identification and dissemination of appropriate protocols within the research community is required to support any realistic expectations of reconfigurable technology solving the performance and versatility requirements of *pervasive handheld computing systems.* The middleware protocols proposed within the context of this thesis are outlined in Chapters Six and Seven.

2.4.1 Middleware Technology Supporting Integral Component Placement

There is a shortage of research focusing upon middleware protocol support for handheld devices incorporating reconfigurable logic. The majority of research projects investigating reconfigurable hardware deployment within portable computers focus upon the physical framework of the device. Examples of this type of research are the Chameleon and Gecko projects as outlined in Section 2.3.1. These research initiatives do not concentrate upon network support for device reconfiguration.

Research dealing with developing middleware infrastructures to support networked reconfigurable hardware based handheld devices is slowly beginning to improve. This can be attributed to both a maturing of reconfigurable logic circuitry (ref. Section 3.1) and to an awareness of the increasing functional demands of portable devices.

An architectural framework for networked reconfigurable handheld devices constructed with EJB[6] technology has been proposed [Nitsch, 03]. Modularity and flexibility are key objectives of the middleware framework. The developed architecture contains a system management API[7] constructed with XML[8]. This

[6] Enterprise Java Bean

[7] Application Program Interface

facilitates services such as application download-on-demand and on-the-fly update of hardware acceleration components.

A Java-based system architecture supporting networked reconfigurable handheld devices has been proposed [Lee, 00]. This framework enables reconfigurable hardware-software based Java applications to be remotely downloaded to the embedded system through a *System Manager* object. The *System Manager* supports two middleware protocols, namely:

- A download protocol catering for the remote installation of reconfigurable bitstreams on an embedded device.

- A system maintenance protocol catering for remote management and testing of the reconfigurable hardware circuitry on the embedded system.

The *System Manager* object uses a simple TCP/IP[9] Java-based socket connection to establish protocol communication between the remote management system and the embedded device.

There have been a number of additional research projects focusing upon Java-based middleware frameworks to support networked reconfigurable handheld devices [Ha, 02], [Ha, 01], and [Guccione, 02]. These middleware frameworks are also constructed with TCP/IP socket connections. The architectures support basic modes of operation such as request and retrieval protocols which allow reconfiguration bitstreams to be retrieved from remote servers.

The Enamorado project is a European Union funded research development focusing on producing a handheld device with the capability of receiving and executing multimedia data streams [Nikolouzou, 04], [Nikolouzou, 04b], and [Kosmatos, 04].

[8] eXtensible Markup Language

[9] Transmission Control Protocol / Internet Protocol

Reconfigurable hardware within the device is employed to process the multimedia content. A primary feature of an Enamorado portable computer will be its capacity to dynamically reconfigure its hardware functionality depending upon media content. Middleware protocols operating over a Java platform support the Enamorado client handheld device and these include:

- Media Content Delivery Protocols
  Media content can be streamed or downloaded to the handheld device depending upon user preferences.

- Application Code Services Protocols
  Reconfigurable hardware-based applications can be downloaded dynamically according to the usage and performance requirements of each user. These applications handle the processing of content from streaming media servers. Protocols are also being developed to allow propagation of upgrade functionality to handheld devices which is initiated from provisioning servers.

The Enamorado project is currently nearing completion with an expected delivery date of December 2005.

2.4.2 Middleware Technology Supporting Networked Resource Placement

The principal benefits of providing a portable computer with a sophisticated middleware framework to access networked reconfigurable hardware resources are:

- Enhanced Handheld Device Performance and Versatility
  A mobile device can offload computationally-intensive tasks to a neighbouring adaptive server.

- Enhanced Utilisation of Reconfigurable Resources
  The resources of a networked adaptive server can be shared by many handheld devices.

The middleware infrastructure supporting the handheld device to offload computational requests to neighbouring adaptive servers should also facilitate dynamic allocation of computational tasks amongst adaptive servers. Runtime binding of application requests with networked reconfigurable resources can improve utilisation of server-based resources by handheld devices.

This is relevant as reconfigurable resources within adaptive servers are costly commodities. Effective utilisation helps ensure satisfactory return on investment as well as the highest possible performance and versatility gains for client handheld devices.

This constitutes an economic and system performance challenge for the middleware technology supporting portable computers. Communication protocols are required to ensure a fair workload distribution amongst all adaptive servers. This helps establish a load-balanced network which in turn provides higher quality of service to all client mobile devices.

This also helps avoid both system bottlenecks and under-utilisation of resources. The quality of service experienced by portable computers is also enhanced as their computational task is dynamically bound to an adaptive server with the lowest workload.

There have been a number of research efforts which have investigated middleware solutions to support client systems in utilising adaptive servers. An attempt to establish ubiquitous access to remote reconfigurable hardware has been outlined [Indrusiak, 03]. Their objective is to allow a network of reconfigurable hardware modules be accessible transparently by client applications.

This entails facilitating client application usage of reconfigurable hardware modules regardless of physical location. This is achieved by raising the abstraction level of the integration architecture for reconfigurable modules and computer systems through the employment of a Jini-based middleware solution [Arnold, 99].

Middleware capable of discovering and exploiting under-utilised computing nodes containing reconfigurable FPGA-based accelerator boards has been developed [Gaj, 03]. This entailed extending an off-the-shelf job management system to facilitate sharing of remote reconfigurable resources. The framework facilitates scheduling of client requests for usage of remote reconfigurable hardware.

A Middleware solution has been constructed to facilitate remote execution of both reconfigurable hardware and software based computational requests [Smith King, 01]. Their research introduces a Java-based communication layer called PEP[10] which provides an interface between a client-side application and server-side target implementations. This abstraction layer is beneficial as it allows the dynamic binding of applications to target implementations at runtime depending upon resource usage. The PEP development also facilitates sharing of reconfigurable hardware resources.

An early visionary paper recognised the potential of agent technology as an effective middleware to utilise networked reconfigurable hardware within a distributed medical domain [Mapen, 99]. Their research efforts outline the benefits of agent technology within healthcare environments. The notion of agents facilitating efficient utilisation of networked adaptive servers is also conveyed. Mapen's work is presented at a high theoretical level and does not focus upon the unique characteristics of personal mobile computing.

A commercial venture marketing a middleware methodology which supports management of networked reconfigurable hardware has been launched [Casselman, 02]. Their business strategy is based upon the belief that an effective middleware capable of transmitting bitstreams to networked reconfigurable logic is an essential element for any FPGA-based product strategy. Their bitstream management environment coins the term *intelligent bitstream* to refer to a C++ object which includes all the data necessary for delivery, verification and use of a reconfigurable bitstream on a networked FPGA.

[10] Packet Exchange Platform

The IRL[11] tool is also a commercial venture developed by Xilinx Corporation [Xilinx, 05] consisting of a network management API to dynamically configure and execute remote server-based reconfigurable logic [Jacobson, 00], [Jacobson, 01], [Xilinx, 01]. A conceptual diagram of the elements within a networked system employing the IRL tool is presented in Figure 2.2.

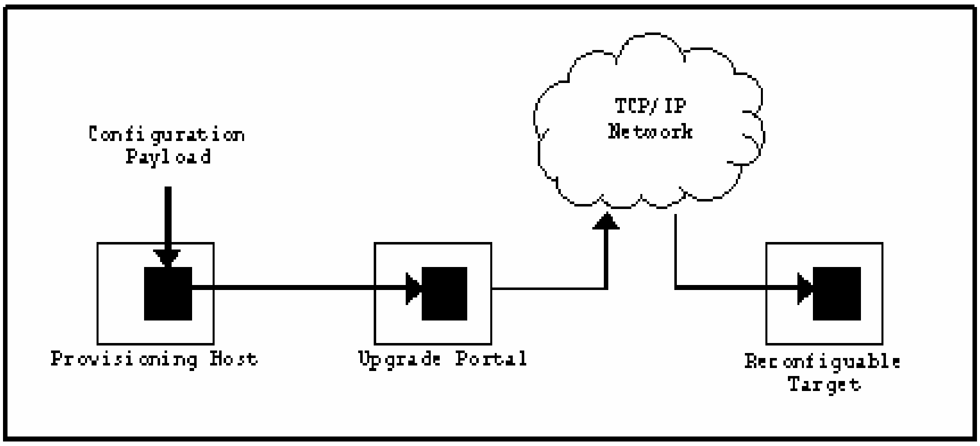


Figure 2.2: Block Diagram of Internet Reconfigurable Logic System

IRL provides middleware protocols to upgrade adaptive server functionality through upgrades and fixes. The middleware is built with Java technology and is modelled upon a traditional client server framework. Xilinx developed the JBits API to support the IRL management tool. An overview of the JBits framework is presented in Section 3.6.1.2. IRL provides a basic infrastructure supporting reconfigurable hardware-based adaptive servers.

Xilinx Corporation together with Wind River Systems [Wind, 05] has also developed an object-oriented framework to support the management of reconfigurable hardware in an upgradeable infrastructure [Pave, 05]. Their PAVE[12] framework consists of a communication and configuration management API. The elements of the PAVE framework are shown in Figure 2.3.

[11] Internet Reconfigurable Logic

[12] PLD API VxWorks Embedded

The components of the PAVE framework are a collection of C++ classes and object models that help to abstract and conceptualise reconfigurable hardware-based applications.

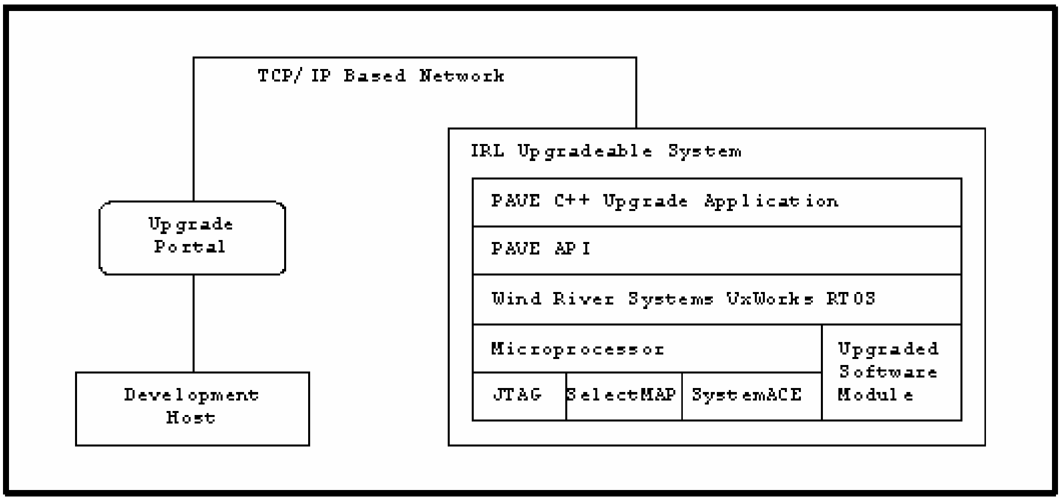


Figure 2.3: Block Diagram of the PAVE Framework

PAVE treats the programmable hardware as an object within the system which encourages applications written with this API to be object-oriented, modular and upgradeable. The PAVE framework introduces a coherent approach to the development of applications for reconfigurable hardware-based server systems.

The key objective of the RARE[13] project is to construct a framework enabling adaptive servers to deliver functionality equivalent to application servers [Doss, 99]. An illustration of the RARE framework is presented in Figure 2.4. RARE enables client programs to attach to an adaptive server through a Java-based interface to remotely utilise available reconfigurable resources. Java technologies utilised within the methodology include JNI[14] and RMI[15].

RARE presents a good basic framework for combining network technologies and reconfigurable computing to provide a distributed adaptive environment. However, there are certain disadvantages in the methodological approach.

[13] Remote Adaptive computing Resource Environment

[14] Java Native Interface

[15] Remote Method Invocation

Scalability could prove a headache as substantial development would be required to facilitate fair workload distribution amongst a number of adaptive servers.

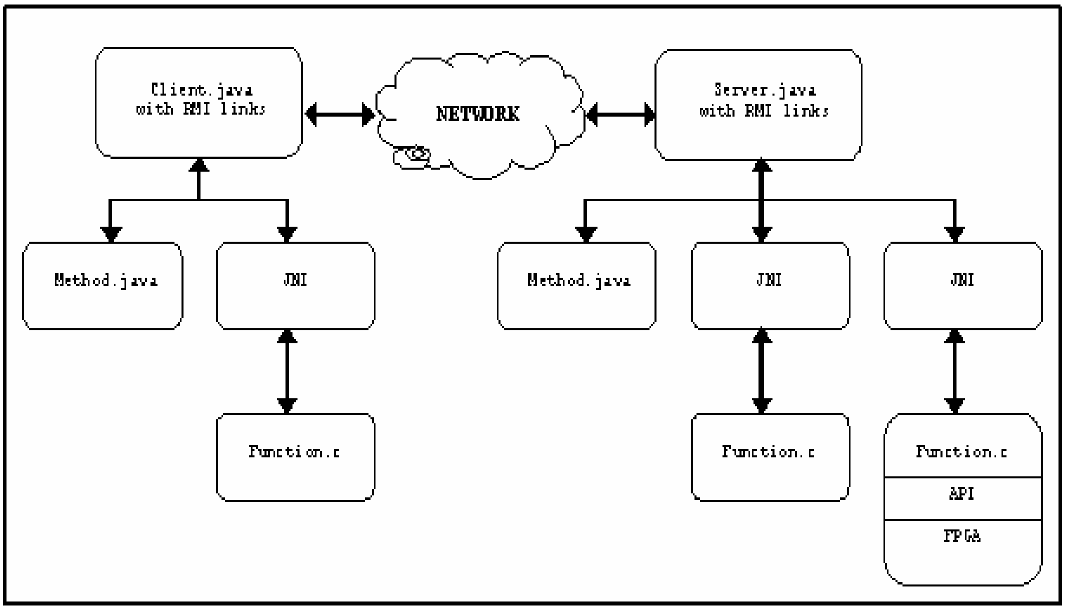


Figure 2.4: Block Diagram of the RARE Framework

Additional Java-based middleware solutions enabling dynamic interaction with remote reconfigurable hardware resources within adaptive servers have been proposed by Moseley [Moseley, 02] and Agarwal [Agarwal, 94]. There have also been a number of research efforts focusing upon middleware frameworks to support reconfigurable hardware usage over the Internet [Cret, 02], [Wirthlin, 02].

2.5 Mobile Device Technology within Telemedicine Environments

The prototype systems constructed within the context of this thesis are targeted towards mobile telemedicine environments. These experimental prototypes encapsulate middleware communication protocols which seek to enhance utilisation of reconfigurable hardware components within handheld medical device environments.

The primary reasoning for placing these test-case systems within the medical field was recognition of the necessity to equip healthcare professionals with high-performance and multi-functional portable computers. This is especially prevalent as:

- Medical practitioners are nomadic in their work-day practices and handheld devices provide them with the most-likely mechanism of interacting with hospital networking infrastructures.
- Healthcare professionals require compute-intensive functionality from their portable computers e.g. image processing, cryptography services, etc.

Mobile device technology will play a significant role within future medical environments. Portable computers within medical environments can enhance the productivity and efficiency of healthcare professionals. They are a key component in realising the future telemedicine vision of ubiquitous healthcare.

The concept of ubiquitous healthcare refers to the provision of any type of IT-related health service through mobile computing devices [Kirn, 02]. Handheld devices are capable of providing a range of health service applications to medical practitioners.

Enabling these nomadic professionals with access to efficient medical tools at locations of their choice enhances their productivity levels and can also improve the accuracy of their patient diagnosis. These portable devices can help provide greater levels of patient care and help the healthcare system exploit ubiquitous computing capabilities.

There have been a number of research efforts investigating potential benefits and possible strategies for deploying portable devices within a medical environment. The major focus of this work has been an effort to enable medical professionals' access, manipulate and analyse patient records whilst on the move.

The clear benefits associated with providing wireless handheld access to clinical patient records have been recognised [Ancona, 01]. This work compared an electronic patient-based record system accessible on portable computers with a traditional paper-based system. The study showed electronic records provided a clear improvement in the productivity of healthcare professionals in comparison with the conventional paper-based system. The potential for minimising errors through utilising the wireless-based system was also recognised.

A web-based telemedicine architecture facilitating wireless access to electronic patient records on a portable device has been proposed [Lamberti, 02]. This approach utilises Java, XML and XSL[16] technologies. The web browser of a mobile device is incorporated as the visual interface. The inherent benefit of using the Internet as a communication medium is its ability to operate independently of the client hardware/software architecture.

The ability to access images of patient scans everywhere and anytime on mobile hardware has also been investigated. This type of wireless application was identified as beneficial for medical practitioners when performing routine diagnostics [Kroll, 02]. This feasibility study examined applications developed for viewing and analysing DICOM[17] image and waveform objects on handheld devices.

Their conclusions recognised the importance of developing handheld applications that prioritised intelligent interaction. This approach helps minimise drawbacks imposed by the physical constraints of a mobile device. The work also focused on the computational power shortcomings of mobile devices. A traditional client-server framework was implemented to enable handheld devices to offload compute-intensive tasks to neighbouring servers.

2.5.1 Reconfigurable Hardware and Mobile Medical Devices

Computational performance and power consumption of mobile medical devices are related issues of increasing performance within the telemedicine community. Reconfigurable computing is considered by many as having the potential to address these concerns.

The cost, size and power consumption of medical devices is shown to be reduced through utilising a framework consisting of reconfigurable technology and efficient communication techniques [Martel, 00]. This work shows that large

[16] Extensible Stylesheet Language

[17] Digital Imaging and Communications in Medicine

repetitive tasks implemented in reconfigurable hardware rather than software can help yield substantial improvements in power and performance.

The authors conclude increased performance and flexibility for future bio-instruments and medical devices will continue to be enhanced as reconfigurable technology becomes increasingly prevalent and communication structures become more advanced.

2.5.2 Agent Technology and Mobile Medical Devices

The field of agent technology is viewed as a highly suitable paradigm and inter-communication infrastructure for the analysis and design of mobile telemedicine systems [Della Mea, 01]. The agent paradigm is viewed as a vast improvement to the traditional client-server approach for developing complex telemedicine systems. Such systems can be defined as communities of interacting entities that aim to support collaboration and resource sharing in a medical environment.

This observation is especially prevalent for mobile telemedicine systems which have continuously appearing and disappearing components within their distributed network. Agent-based frameworks are recognised as being capable of coordinating distributed decision-making processes within these medical-based environments [Huang, 95]. Agents are seen as abstractions within these hospital environments that can hide the complexities associated with collaborative work practices.

Agent middleware is also recognised as a key factor in facilitating the ambient intelligent vision within future healthcare environments [Rodriguez, 04b]. Ambient intelligence is a research paradigm that aims to empower individuals within digital environments through awareness of their context and activity whilst having the capability of displaying sensitivity and reactivity to their needs.

Previous research recognised the potential of agent technology as an effective middleware to utilise reconfigurable hardware within a distributed medical domain [Mapen, 99]. This earlier work outlines the advantages of using agent technology within healthcare environments and also details its ability to efficiently exploit networked adaptive servers.

2.6 Summary

Reconfigurable computing and agent technology have key roles to play in the future growth of mobile telemedicine systems. The technical attributes of each of these research paradigms could be considered essential to the field of mobile telemedicine and to the concept of ubiquitous computing within healthcare.

Future handheld computers within the medical field will clearly need to be high-performance, multi-functional devices that are intelligently utilised so that they are capable of executing a broad range of compute-intensive applications [O' Sullivan, 05e].

The deployment strategies presented within this thesis highlight a technological solution for delivering such handheld devices. These deployment strategies consist of medical-based network environments that incorporate a dynamic physical architecture (i.e. reconfigurable hardware) as well as a sophisticated network support infrastructure (i.e. context-aware agent middleware).

The deployment scenarios are by-products of the proposed two-pronged methodological approach. This methodology recommends agent computing and reconfigurable logic as the *middleware technology* and *hardware resources* elements of the *Pervasive Handheld Axis* model. These technologies will help *pervasive handheld computing systems* within medical environments meet their future objectives.

CHAPTER 3

Reconfigurable Computing within Mobile Environments

3.1 Introduction

The reconfigurable computing concept was first presented by Gerald Estrin in 1963 [Estrin, 63]. However, it is only in the last number of years that reconfigurable hardware has begun to come of age in terms of the potential of integrating the technology into the environment of a handheld device i.e. either as an integral component of a portable computer or as a technology encapsulated within surrounding adaptive servers.

Lower financial costs for reconfigurable hardware units as well as vast increases in their gate densities[18] have been the principal driving factors in the growth of reconfigurable computing systems within handheld device environments. The primary benefits of incorporating reconfigurable logic within handheld computing systems are that it provides mechanisms to increase device performance and system functionality [Guccione, 01]. These improvements are delivered with efficient consumption of power resources. For example, a reconfigurable hardware-based implementation of an AES[19] encryption algorithm reported a 96.7% improvement in power efficiency in comparison with a general purpose processor solution [Verbauwhede, 02]. The reconfigurable hardware-based platform also executed the AES algorithm 5% quicker than the general-purpose processor implementation. An appraisal of the performance capabilities of reconfigurable hardware is presented in Section 3.7.

Underlying these power efficiency and performance improvements is the understanding that individual functions or complete systems realised with hardware execute faster and utilise less power than software-based

[18] Xilinx (http://www.xilinx.com), for one, has confirmed the construction of a reconfigurable hardware device with one billion transistors [Burger, 97].

[19] Advanced Encryption Standard

implementations [Prophet, 04]. Essentially, the key feature of reconfigurable-based computing is that it provides the capability to perform computations in hardware to increase performance, while retaining much of the flexibility of a software solution [Compton, 02].

It is important to note that most computationally intensive applications expend 90% of their execution time within only 10% of their code [Hennessy, 96]. The basic instructions within these 10% code blocks naturally differ from application to application. These observations make the idea of a fast general-purpose central processing unit appear inconsistent. A custom computing machine or a reconfigurable computer allowing for customisation of each application is a solution to this contradiction of delivering general-purpose computing with high-performance processing.

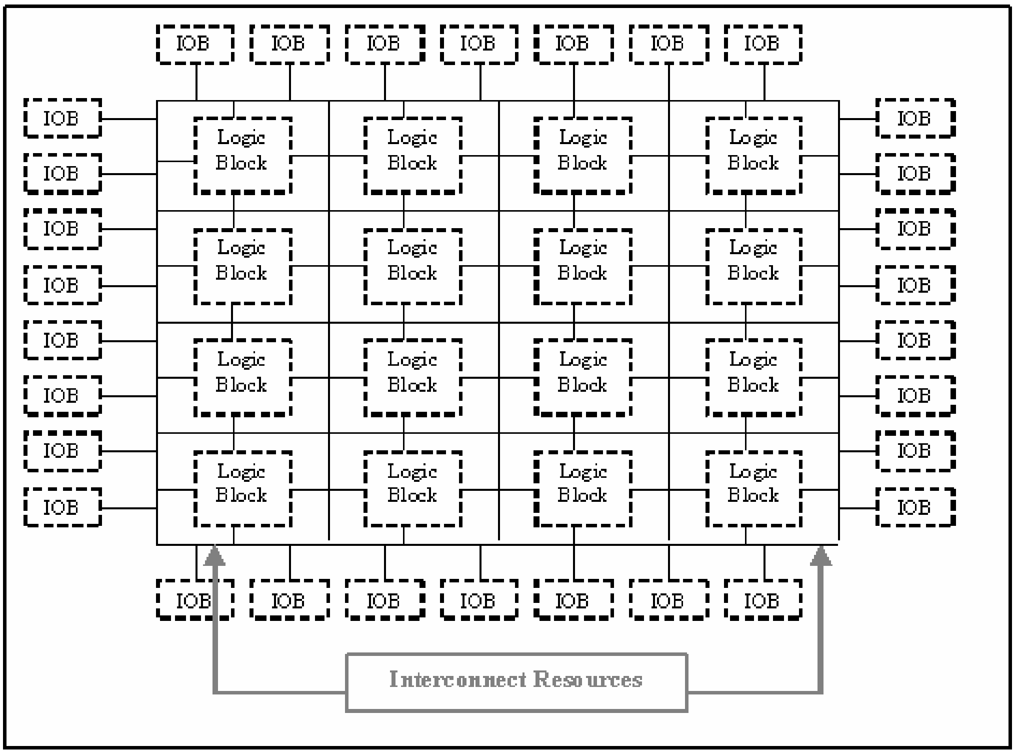


Figure 3.1: FPGA Internal (Island-Style) Architecture

An enabling technology of these custom computing machines is the FPGA[20]. FPGAs were invented by Xilinx Corporation in 1984 [Xilinx, 02]. FPGAs are

[20] Field Programmable Gate Array

used to accelerate algorithm execution by mapping compute-intensive calculations to the reconfigurable substrate. A detailed examination of the individual components within an FPGA is presented within Section 3.2.

Basically, reconfigurable computing involves manipulation of the logic within an FPGA during system runtime. In essence, the design of the hardware may change in response to the demands placed upon the system whilst it is running. The FPGA can act as an execution engine for a variety of different hardware functions and these may operate in parallel or in serial mode.

3.2 FPGA Architecture

The internal architecture of an FPGA is presented in Figure 3.1 and it consists of the following components:

- Configurable Logic Blocks (CLB)

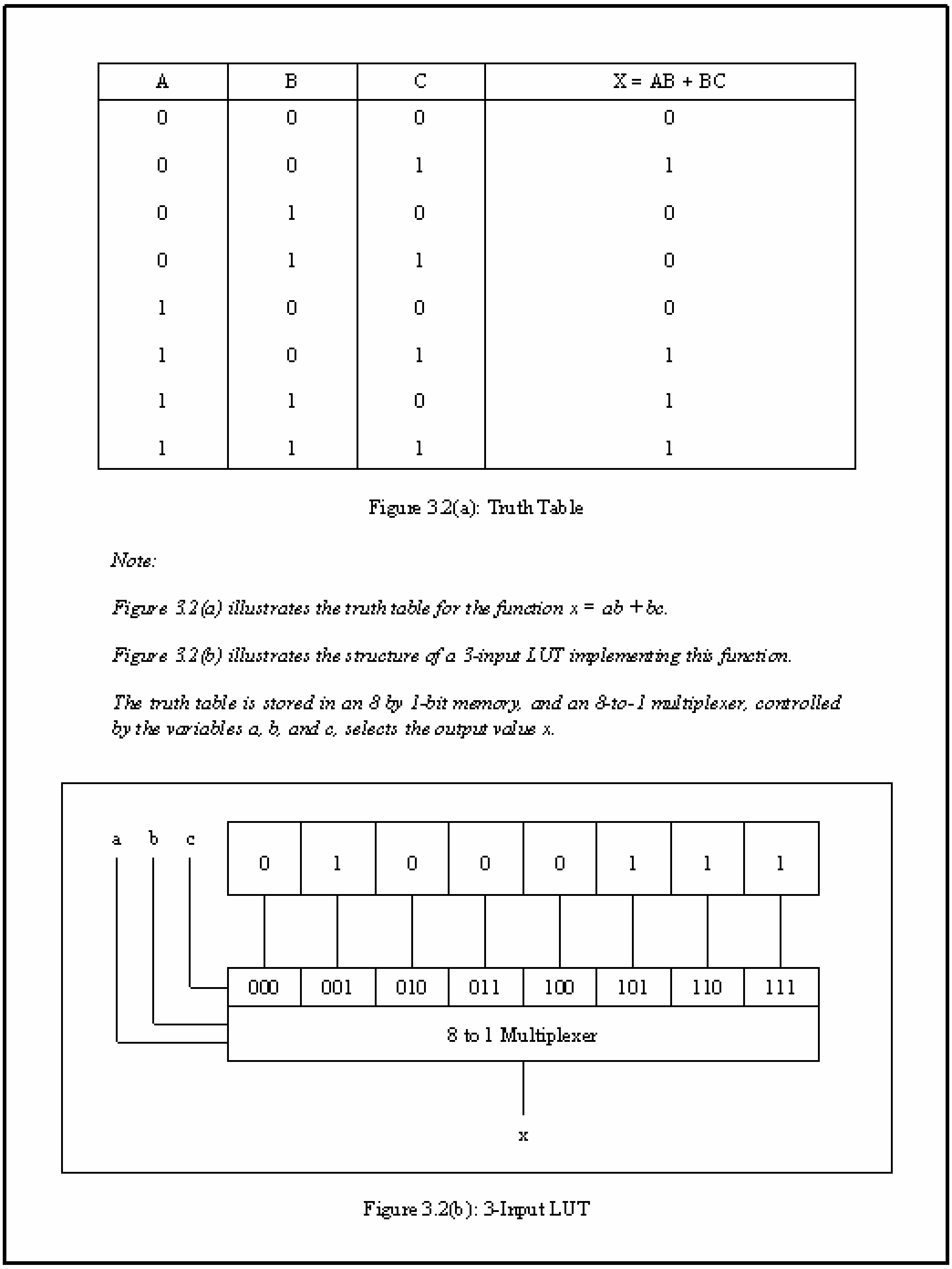
  A CLB is a group of functional elements (look-up tables, multiplexers and flip flops) used in constructing logic. A look-up table (LUT) logic gate has chameleon-like capability to emulate any other possible logic gate. An *n-input* LUT is an *n-address* memory used to store the possible values of an n-input boolean function. With an *n-input* LUT, it is possible to implement any function with *n* variables. The values of the function for any combination of the *n* variables is computed and stored in the LUT [Francis, 92]. An example of this flexibility is presented within Figure 3.2.

- Input-Output Blocks (IOB)

  An IOB provides an interface between an outside package pin and an internal signal line.

- Programmable Interconnect Resources (PIP)

  Internal connections are composed of metal wires with programmable switching points to implement the desired routing. Programmable routing is utilised to interconnect the inputs and outputs of each CLB and IOB onto the appropriate networks [Xilinx, 03]. Each type of interconnect is distinguishable by the relative length of the segment e.g. single-length lines, double-length lines, and long lines.



A	B	C	X = AB + BC
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 3.2(a): Truth Table

Note:

Figure 3.2(a) illustrates the truth table for the function x = ab + bc.

Figure 3.2(b) illustrates the structure of a 3-input LUT implementing this function.

The truth table is stored in an 8 by 1-bit memory, and an 8-to-1 multiplexer, controlled by the variables a, b, and c, selects the output value x.

Figure 3.2(b): 3-Input LUT

Figure 3.2: Example of Look-up Table Flexibility

A user design is implemented within an FPGA by configuring the CLB elements to determine the digital logic and by configuring the routing to determine the internal connections of the FPGA. An example schematic of a Xilinx Virtex CLB slice is shown in Figure 3.3.

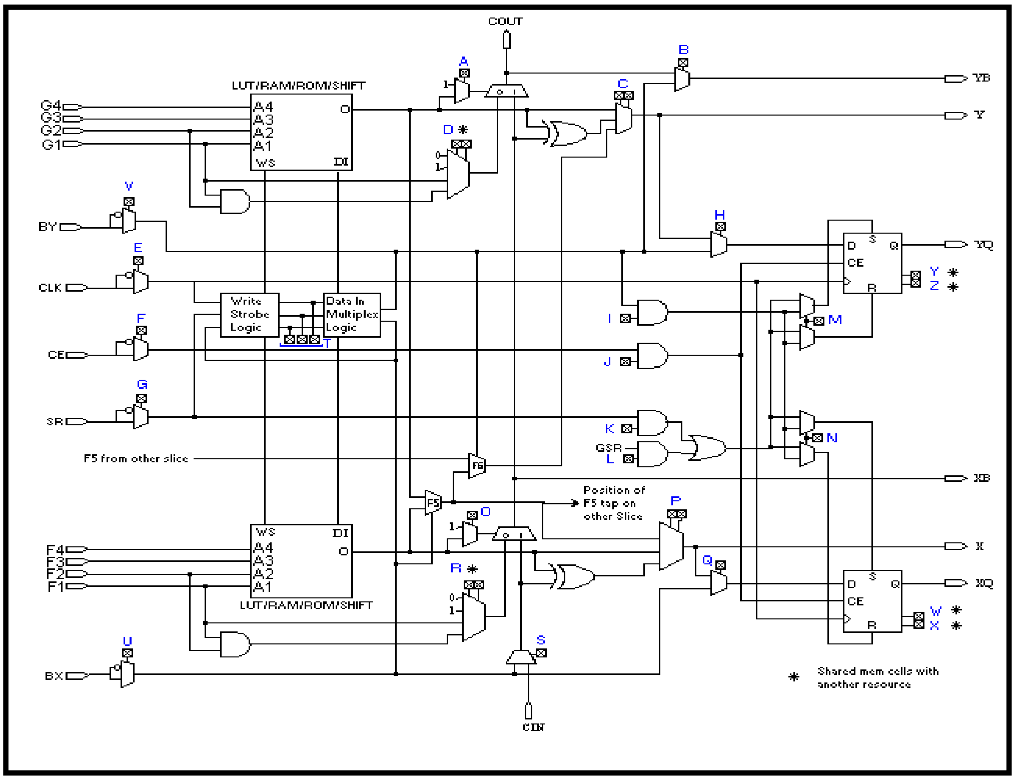


Figure 3.3: Xilinx Virtex Device CLB Slice

This CLB has two 4-input LUT(s), D-type flip-flops and some form of fast carry logic. The general routing allows data to be passed to or received from other CLBs. The input multiplexers allow wires in the general routing to pass data to the slices, while the output multiplexers allow the slices to pass data to wires in the general routing. The Xilinx Virtex FPGA is the reconfigurable hardware employed within the prototype systems developed and presented within this thesis.

3.3 Reconfiguration Modes

Reconfigurable computing has several possible execution models which can be characterised by the context of their architectural design. This classification divides the design models into three programmability classes based upon their number of configurations and the time in which reconfiguration takes place [Adario, 99]. These design models can be described as follows:

- Static Design

  This type of circuit design maintains a single configuration which is never modified. An architectural flow diagram of this model is presented in Figure 3.4 (a). This design does not exploit the reconfigurable aspects of the hardware. The principal benefit of this design model is the high-level of flexibility available during the prototyping phase.

- Statically Reconfigurable Design

  This type of circuit design has several configurations with any reconfiguration process occurring at the end of a processing task. An architectural flow diagram illustrating this type of model is presented in Figure 3.4 (b). This design model may also be classified as run-time reconfiguration. The programmable device within this model is better utilised through dynamic modification of hardware resources. Reconfigurable computing systems encompassing this design model use an FPGA like a demand-paged resource. This delivers a *virtual hardware* methodology which is analogous to the concept of *virtual memory* [Hauck, 98]. Image processing applications are well-suited to this design model. These applications often consist of a pipeline of image filter components. Each component applies a different image processing algorithm and may be represented as a separate configuration mapping for the reconfiguration logic [Quinn, 03].

- Dynamically Reconfigurable Design

  This type of circuit design has multiple configurations with any reconfiguration processes occurring during runtime whilst other hardware processes may be executing. An architectural flow diagram illustrating this type of model is presented in Figure 3.4 (c). This design is more efficient in its use of reconfigurable resources but is also more complex in its design. The design model may also be classified as partial run-time reconfiguration. The implementation may use either a partially programmable device (e.g. Xilinx Virtex FPGA) or a set of conventional programmable devices (multi-FPGA architecture).

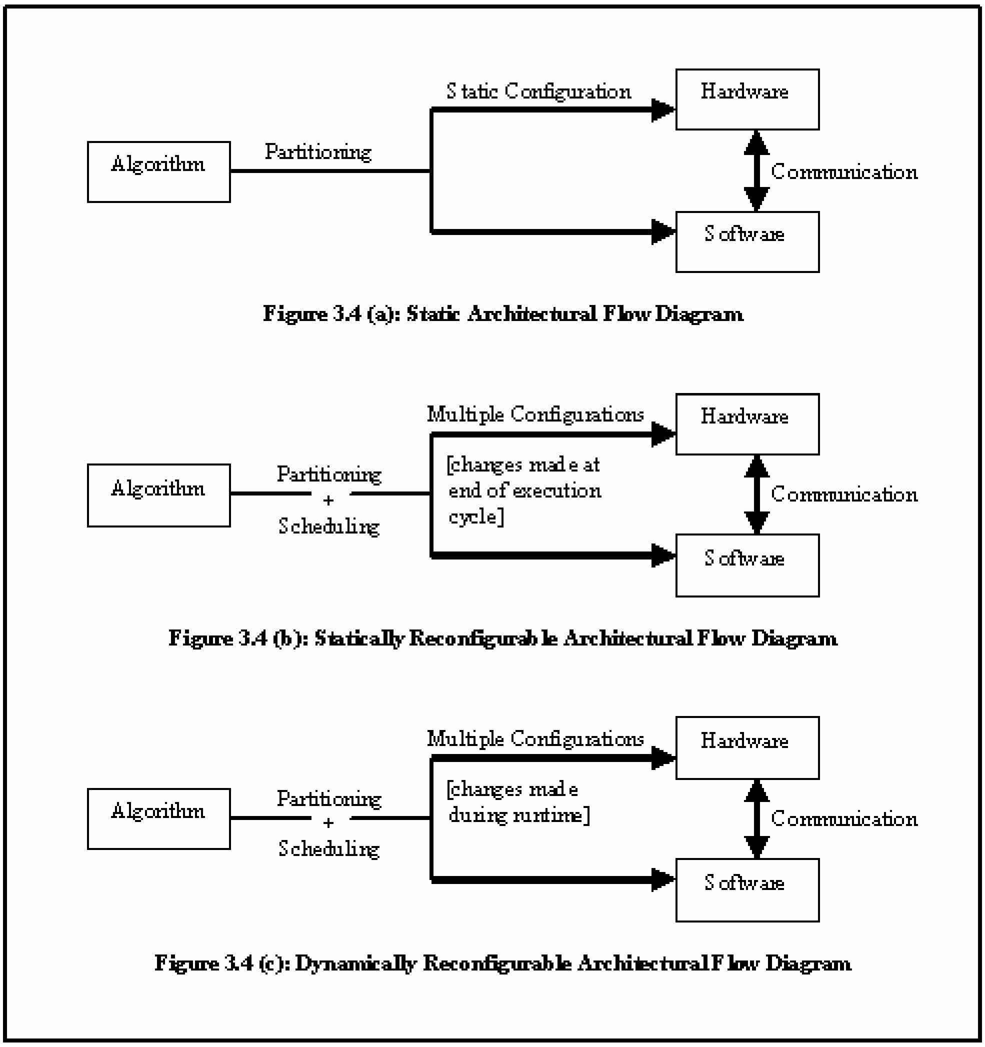


Figure 3.4 (a): Static Architectural Flow Diagram

Figure 3.4 (b): Statically Reconfigurable Architectural Flow Diagram

Figure 3.4 (c): Dynamically Reconfigurable Architectural Flow Diagram

Figure 3.4: Architectural Flow Diagrams of Reconfiguration Modes

The prototypes developed within this thesis to evaluate *pervasive handheld computing systems* are within the category of *statically reconfigurable design*. However, the software tools and device drivers (ref. Section 3.5) utilised allows the prototype systems to facilitate the *dynamically reconfigurable design* model.

3.4 FPGA Design Flow

Reconfigurable systems are fundamentally hardware-software based embedded systems that have the added capability to modify reconfigurable logic resources dynamically. The design flow methodology employed in the construction of prototypes to evaluate *pervasive handheld computing systems* is presented within Figure 3.5.

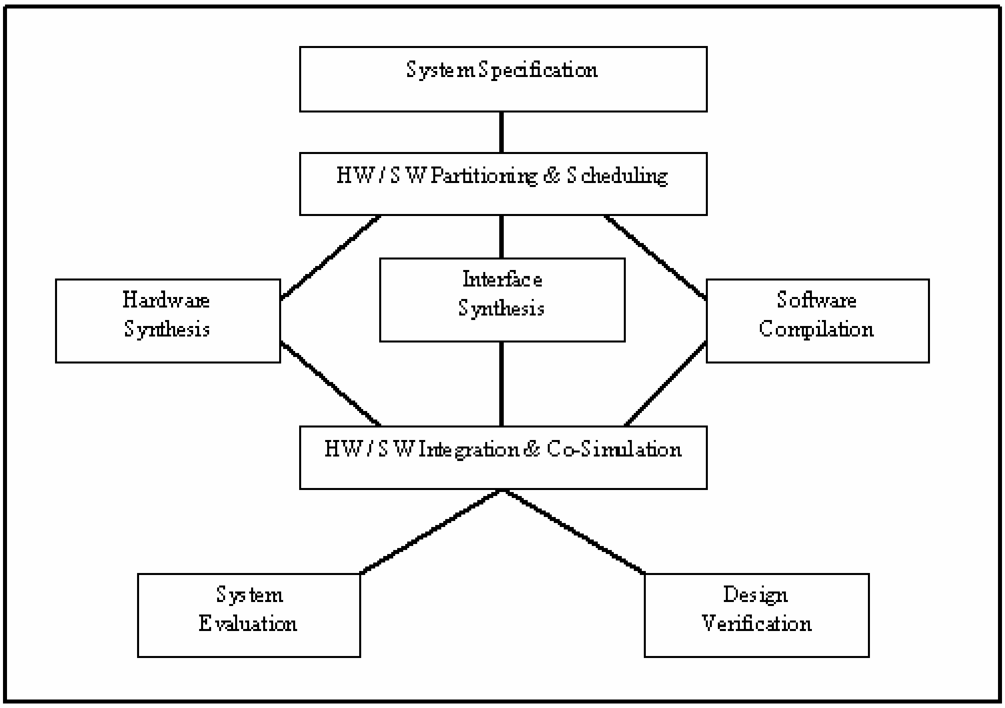


Figure 3.5: Design Flow Methodology for Pervasive Handheld Computing Systems

Each stage within the design flow methodology can be described as follows:

- System Specification

  This stage consists of a detailed analysis of the hardware and software elements of an application and the relationships between them.

- Partitioning

  This concerns the process of deciding, for each subsystem, whether the required functionality is more advantageously implemented in hardware or software. The goal is to achieve a partitioning that delivers the required performance within system constraints (e.g. size, power, processing speed).

- Scheduling

  The scheduling process enables hardware and software resource sharing. The task of scheduling is to determine task assignment and execution order on the processing elements (processor or FPGA) whilst respecting time constraints.

- Hardware Synthesis

  Hardware synthesis is the process of converting a high-level language / hardware description language (HDL) / schematic capture representing the hardware domain of the system specification to the relevant vendor-specific FPGA device bitstream format. The hardware synthesis tools utilised in delivering *pervasive handheld computing system* prototypes are outlined in Section 3.6.

- Software Compilation

  This stage involves using compilers to generate processor specific machine code representing the software domain of the system specification.

- Interface Synthesis

  This is a process of defining the interface between the hardware and software components and synchronising interaction between them. The interface requires access to driver libraries to facilitate communication to peripheral devices. A detailed description of the interface employed within the *pervasive handheld computing system* prototypes is presented in Section 3.6.

- System Evaluation

  This stage involves ensuring deliverable product meets original requirements of system specification.

- Design Verification

  The design verification phase entails modular testing of hardware and software components to ensure functional compliance with requirements. The stage also includes verification of the complete system to ensure all necessary constraints are met.

3.5 Partitioning and Scheduling Issues

There are varying levels of complexity associated with partitioning and scheduling of application solutions. The degree of complexity is dependent on the level of hardware reconfiguration supported by the reconfigurable system (i.e. static design, statically reconfigurable design, or dynamically reconfigurable design).

Partitioning and scheduling constraints become increasing difficult to obey as the degree of reconfiguration flexibility improves. This is primarily attributable to the issues of configuration latency and partial reconfiguration [Mei, 00]. The configuration time associated with the execution of a task is dependent on the preceding and subsequent tasks contained within the schedule of operation for the dynamic reconfigurable system. Additionally, partial reconfiguration changes the operation of a reconfigurable system from acting in a sequential processing mode to functioning like a resource pool. This complicates the process of scheduling tasks within an FPGA such that it becomes a constrained placement problem.

This has led to a number of research efforts which have attempted to address automatic partitioning and scheduling for reconfigurable architectures. The Garp and Nimble compilers allow for automatic partitioning of C-type specifications for FPGA-based platforms [Li, 00], [Hauser, 98], [Harr, 00]. These compilers facilitate instruction-level parallelism but are not suited to task-level parallelism. Crusade and Cords are systems capable of synthesising tasks across a distributed network of multiple dynamically reconfigurable FPGAs [Dave, 99], [Dick, 98].

The partitioning and scheduling of tasks within the prototypes developed to demonstrate *pervasive handheld computing systems* are pre-determined manually with the aid of system-level designs. Future research will expand upon this approach to incorporate automated partitioning and scheduling techniques.

3.6 Reconfigurable Hardware Development Tools

The reconfigurable hardware development tools employed within prototype development of mobile medical computing devices presented within this thesis are primarily concerned with the processes of hardware synthesis and interface synthesis.

3.6.1 Hardware Synthesis Development Tools

Hardware synthesis is the process of converting the hardware domain of a system specification (i.e. a high-level language, a hardware description language or a schematic capture) to a relevant vendor-specific FPGA device bitstream format [Ma, 03]. Hardware modules used within the *pervasive handheld computing systems* synthesis process were defined with VHDL[21], JBits and Handel-C. These hardware development languages are utilised to describe hardware modules within problem specifications. The Xilinx IDE[22] toolkit was employed to convert hardware module designs into board-specific bitstreams [Xilinx, 05b].

[21] VHSIC Hardware Description Language

[22] Integrated Development Environment

3.6.1.1 VHDL

VHDL is a language for describing digital electronic systems [Ashenden, 98]. It is a well-established and standard design language. VHDL has evolved to cover multiple levels of abstraction ranging from structural netlist to RTL[23] to behavioural [Sweeney, 02]. These levels of abstraction are illustrated within Figure 3.6 [Spiegel, 01]:

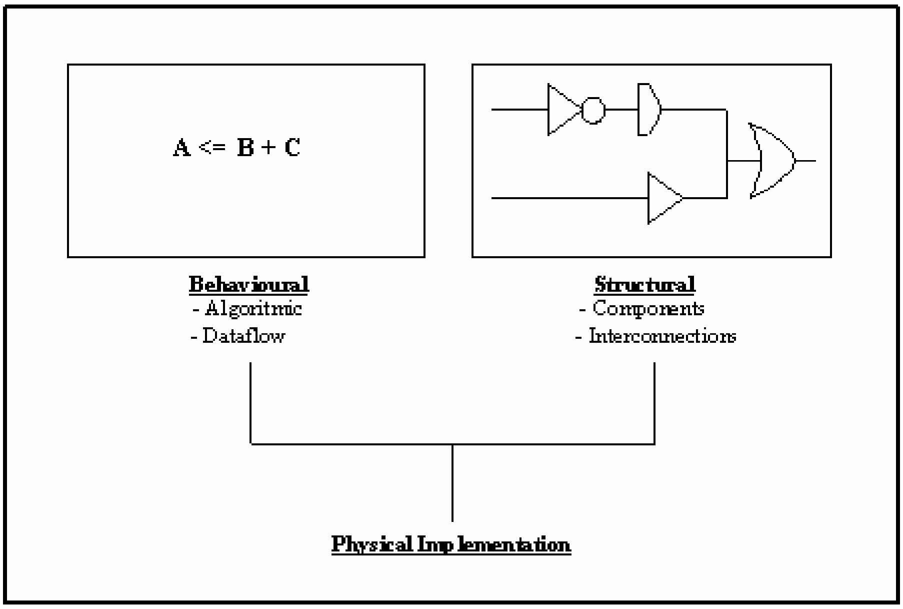


Figure 3.6: Abstraction Levels

VHDL was employed initially as a hardware synthesis tool in the development of the medical-based prototype systems. A VHDL program implementing the DES[24] encryption algorithm for incorporation within the medical-based prototype systems was constructed. The purpose of this initial encryption algorithm was to secure data transfer of confidential patient records within wireless-based telemedicine environments.

The steep learning curve associated with VHDL for developers with software engineering backgrounds [Neff, 03] led to an exploration of other hardware development tools and to an eventual preference to adopt Handel-C.

[23] Register Transfer Level

[24] Data Encryption Standard

3.6.1.2 JBITS

JBits is a Java-based API developed by Xilinx Corporation allowing creation and dynamic modification of their FPGA circuitry [Guccione, 99]. JBits consists of a set of Java classes that provide designers with access to the programmable resources of Xilinx Virtex devices.

The interface allows hardware circuits on this range of FPGAs to be designed, modified and dynamically reconfigured. JBits enables access to look up table circuitry within the configurable logic blocks of an FPGA. The API also facilitates manipulation of the routing resources of the Xilinx Virtex range of devices [Jbits, 05].

At the heart of JBits are four functions as shown in Figure 3.7. The first two allow configuration bitstreams to be read and written. The third function allows the state of a programmable resource to be queried, whilst the fourth function allows the state of a programmable resource to be set to a defined value. The rest of the JBits API is a series of constants defining each of the programmable resources within the device and the value they can be set to. JBits thus hides the proprietary nature of the configuration bitstream whilst still allowing full read and write access to all programmable resources [James-Roxby, 00].

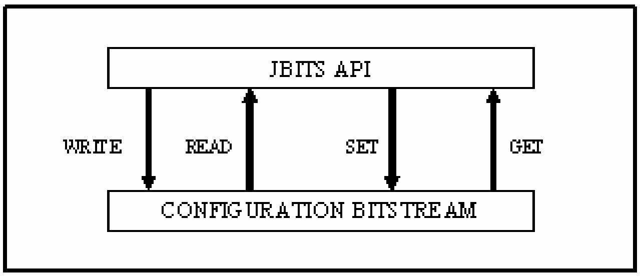


Figure 3.7: JBits API Functionality

The JBits API was originally developed as an enabling technology for runtime reconfiguration but it can also be used to produce traditional static design bitstream files for Virtex FPGAs.

The design flow for a static design is shown in Figure 3.8. In this flow, the design is specified in a Java program using the JBits API and/or JBits cores. The application takes a bitstream file as an input (e.g. null bitstream) and extracts the device configuration data.

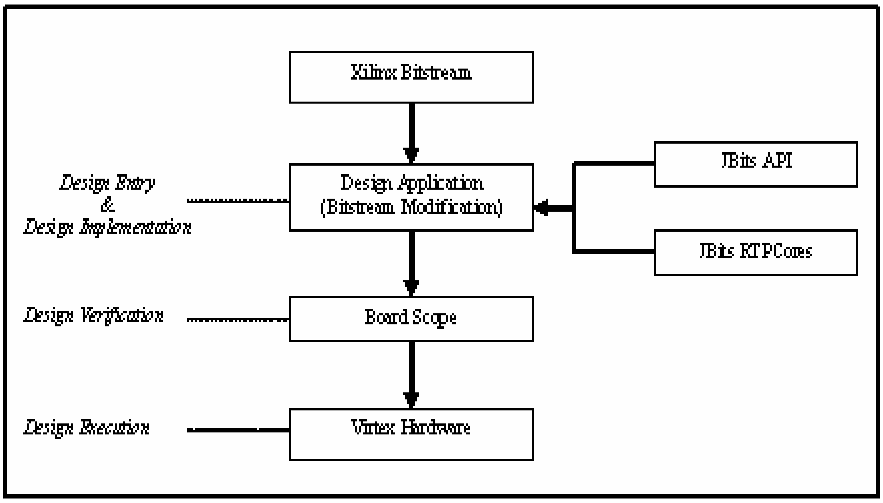


Figure 3.8: Traditional Static Hardware Design using JBits

This data is modified according to the design specified using the JBits API and is output to a bitstream file that will be used to configure the hardware device. An example of a JBits program implementing a grey scale image filter algorithm is presented in Appendix A.1.

3.6.1.3 Handel-C

Handel-C is a hardware-based programming language built upon the syntax of the conventional C programming language [Handel-C, 05]. It provides an alternative design approach to circuit description than HDLs[25] and schematic capture. Handel-C has additional hardware constructs to help gain maximum benefit in performance from a target FPGA.

The advantage of Handel-C is that it provides designers with a higher level of abstraction for hardware design. This benefits software-based engineers as it enables them to design hardware circuitry using familiar development concepts [Todman, 05]. A disadvantage of Handel-C is the potential for loss of performance as it can be argued the design is not as optimised as it would be with HDL / schematic capture approaches [Edwards, 05].

[25] Hardware Description Languages

The Handel-C compiler generates either XNF[26] for Xilinx FPGAs or EDIF[27] for use with non-Xilinx based devices. The design flow for hardware synthesis with Handel-C is shown in Figure 3.9. An example of a Handel-C program implementing an edge detection algorithm is presented in Appendix A.2.

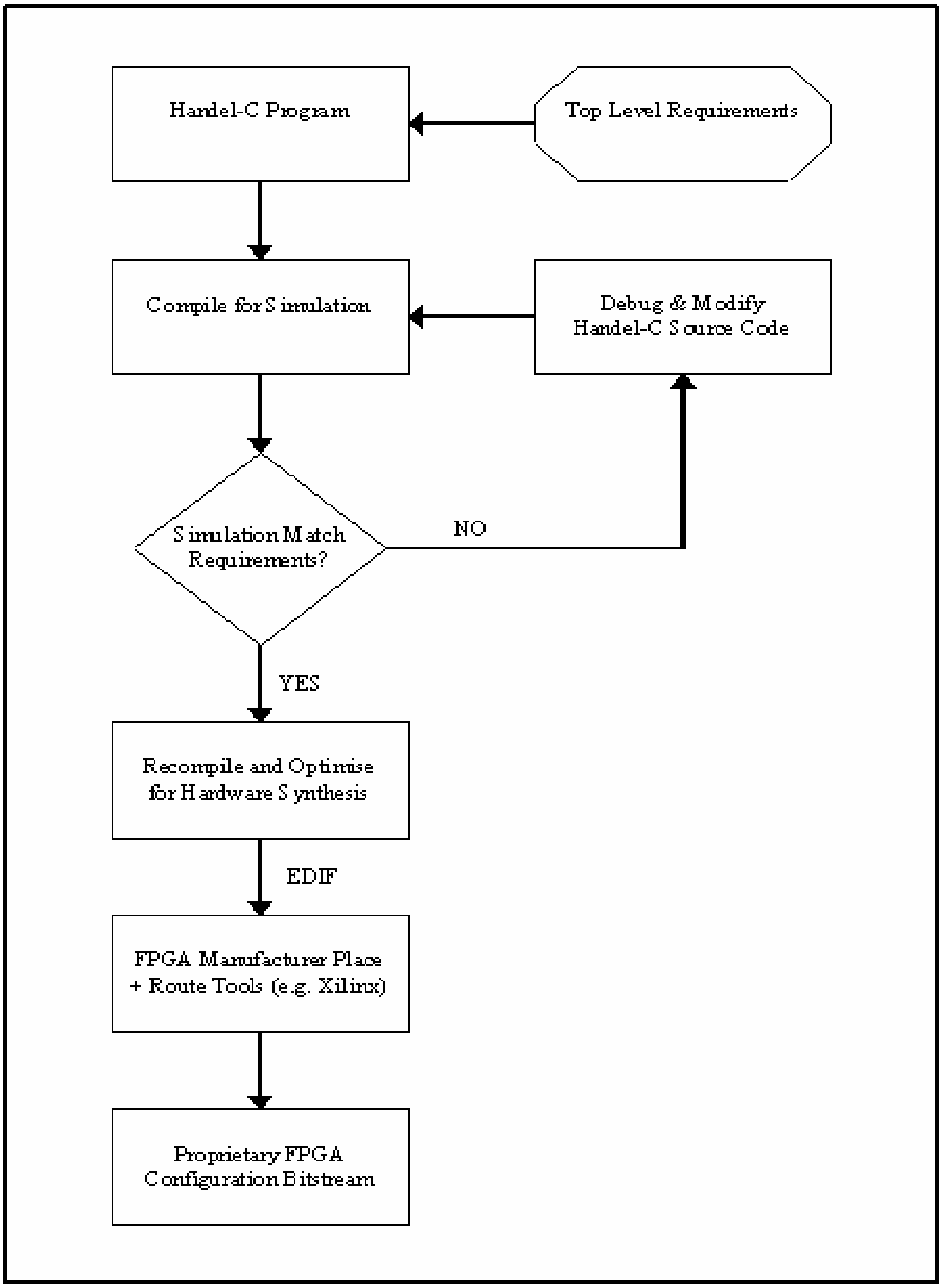


Figure 3.9: Handel-C Design Flow

[26] Xilinx Native Files

[27] Electronic Database Interchange Format

3.6.1.4 Hardware Synthesis Appraisal

Three development methodologies were explored to convert reconfigurable hardware algorithms specified within the medical-based experimental prototype to FPGA bitstream representations. These hardware synthesis techniques are outlined in the sections above. They differ primarily in language specifics, levels of abstraction and in the tools and mechanisms employed to generate executable FPGA code.

VHDL is the oldest and most widely used of the three techniques. It is ideally suited to hardware engineers because effective VHDL development requires informed knowledge of hardware design [Hedberg, 03].

JBits is a favourable mechanism to employ for hardware synthesis as the language is purely Java-based. This feature facilitates the direct integration of hardware-based cores into software agents. This enhances the unity of the design and development process as it allows reconfigurable hardware and software-based agent specifications to be implemented using a single high-level language. Essentially, the hardware and software aspects of an agent's functionality can be constructed uniformly. This can help avoid inter-communication and inter-operability issues arising within large project development teams.

JBits is implemented with a high-level programming language, however its hardware-specific constructs are intensely low level. This necessitates a detailed knowledge of the workings of FPGA architecture and specifically the Xilinx Virtex framework for effective hardware module development. These factors introduce a steep learning curve for effective JBits usage which is further increased with poorly documented resources and a lack of developer support infrastructures[28]. Additionally, JBits is targeted solely towards Xilinx specific FPGAs raising serious concerns regarding the ability to support diverse FPGA manufacturers within a large scale deployment of the medical-based prototype system.

[28] A JBits newsgroup does exist at jbits@yahoogroups.com; however there is no direct support from Xilinx for this group.

Handel-C was the third hardware synthesis methodology explored within the prototype development process. The platform became the favoured choice for application development given its high-level language syntax and associated hardware abstraction. Handel-C is also packaged with device libraries that help to deliver a sophisticated inter-communication infrastructure between the agent platform and the reconfigurable hardware.

3.6.2 Interface Synthesis Tools

The inter-communication layer between the agent platform and the reconfigurable hardware is an essential interface for successful deployment of the medical-based prototype. A Java-based agent executing on a general purpose processor utilises this inter-communication layer to communicate and configure the hardware portion of its code to the reconfigurable logic.

Two separate approaches were identified and examined as potential mechanisms to facilitate inter-communication between an agent platform and reconfigurable hardware, namely:

- JBits XHWIF[29] Interface
- Handel-C Device Drivers

These inter-communication interfaces were explored with regard to their ability to support a Java-based agent framework to effectively exploit neighbouring reconfigurable resources.

3.6.2.1 JBits XHWIF Interface

The JBits XHWIF Interface has its origins within the IRL system design methodology developed by Xilinx Corporation [Xilinx, 01]. IRL facilitates the remote upgrade of hardware and ensures reliability for any modification process. The methodology provides functionality to deliver FPGA bitstreams and software drivers to remote hardware.

[29] Xilinx HardWare InterFace

The communication of updates to remote hardware occurs over a TCP/IP connection. Elements involved in this configuration management operation utilise the JBits XHWIF Interface as can be seen in Figure 3.10. The interface facilitates dynamic reconfiguration of hardware applications allowing circuits to be modified at runtime. Configuration management of FPGA bitstreams is achievable irrespective of the physical location of the target FPGA.

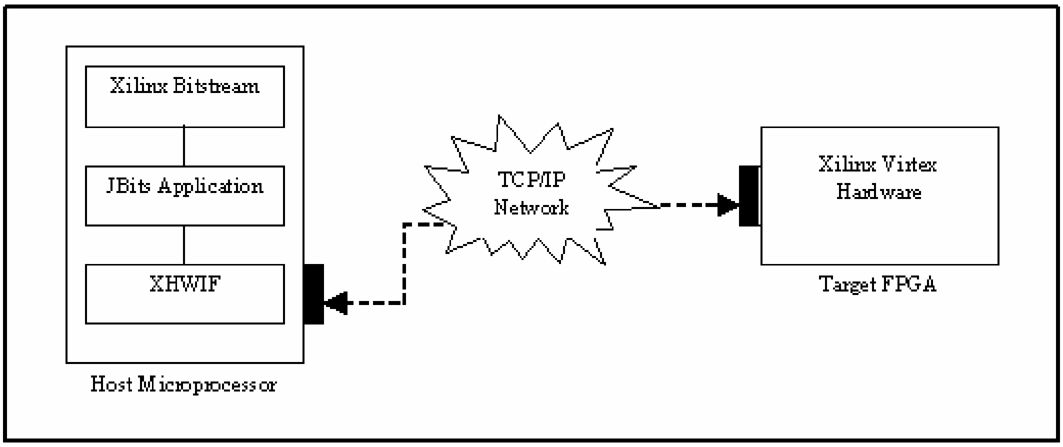


Figure 3.10: JBits XHWIF Interface Deployment

An example of a JBits program utilising the XHWIF interface to initialise and configure an FPGA resource is presented in Appendix A.3. This program illustrates a primary benefit of the XHWIF interface. Commands invoking the interface are written in Java and their operation is controlled directly by the Java Virtual Machine executing on the host machine. This allows for reuse of device driver specific code across multiple platforms irrespective of their operating system environment.

3.6.2.2 Handel-C Device Drivers

The Handel-C development environment is distributed by Celoxica Corporation [Celoxica, 05b]. Their product is packaged with a range of C-based device drivers to facilitate inter-communication between software-based programs executing on a host microprocessor and hardware programs operating on FPGA development boards.

The operations provided by these device drivers include:

- Configuring a bitstream on FPGA substrate
- Passing data for processing to the board
- Controlling algorithm execution on board
- Retrieving data/results from board

The RC200 development board was utilised as the reconfigurable hardware component within the medical-based prototype system. This FPGA device provides a number of peripheral access points to facilitate inter-communication. These include a Serial port, a Parallel port, and an Ethernet access point. These physical peripherals are utilised by the C-based device drivers for communication operations. The device drivers are distributed in DLL[30] format to maintain source code privacy. These DLL's are suitable for Windows 32-bit operating systems. A high-level architectural overview of the Java-based agent platform communicating with reconfigurable hardware using Handel-C device drivers is shown in Figure 3.11.

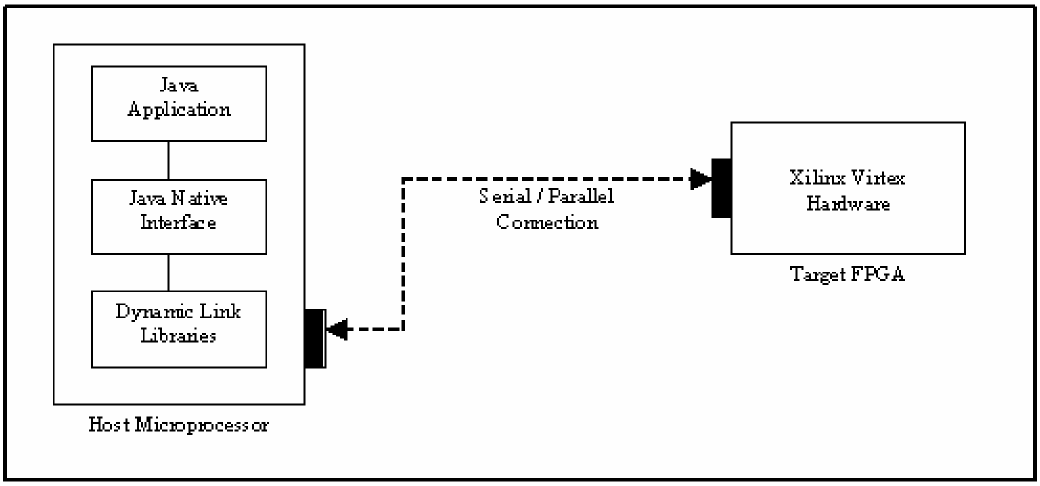


Figure 3.11: Handel-C Device Driver Deployment

The agent platform utilises JNI functionality to establish interoperability with reconfigurable hardware components. JNI enables code written in Java to utilise functionality within native libraries (e.g. C, C++). A consequence of the source code privacy of the DLL's and their ability to only operate with 32-bit compliant microprocessors affects their usage with mobile handhelds. These portable

[30] Dynamic Link Libraries

devices typically operate with an 8/16-bit microprocessor and this raises compatibility problems with the Handel-C device drivers from Celoxica. This issue affects development of a number of medical-based prototype deployment scenarios which contain reconfigurable hardware as an integral component of a portable computer. An account of handheld device usage within these deployment scenarios is outlined in Section 6.4.

An example of a Java program utilising the JNI interface to invoke the Handel-C DLL's to initialise and configure an FPGA resource is presented in Appendix A.4.

3.7 Performance Capabilities of Reconfigurable Computing

Functional decomposition of an application can help determine whether a reconfigurable hardware-based system will be an effective execution platform. In general, complex control sequences and irregular computations are efficiently implemented in software while fixed datapath and data-parallel functionality may be more efficiently executed with reconfigurable logic [Hauck, 98], [Compton, 02].

A range of research projects have documented reconfigurable hardware-based systems delivering high performance solutions. These encompass a wide variety of application areas and have included cryptographic, image processing, and multimedia-based systems.

The Cameron project investigated mapping imaging applications to reconfigurable hardware-based systems [Draper, 02]. The primary motivation for this work was driven by the observation that the comparatively small image processing market deterred manufacturers from ensuring application-specific processors kept pace with advances in processor technology. Consequently, the lifetime of these image-based ASICs was short-lived and left researchers with redundant technology. Fortunately, FPGA speeds and capacities have obeyed Moore's law for the last several years and so this project was aimed at empowering researchers with the tools to generate highly efficient hardware–based image applications. Their results showed FPGA-based implementations of a range of image processing tasks executed between 8 and 800 times faster than high specification Pentium PCs.

Further reconfigurable computing research within the field of image processing and computer vision has focussed on the areas of real-time point tracking [Benedetti, 98], colour-based edge detection [Benitez, 99], and image compression [Hartenstein, 95]. It has also been established that reconfigurable hardware-based implementations of imaging algorithms have occasionally delivered the highest performance solution. This has been the case with an FPGA-based digital image processing system called DRIP[31] which delivered binary and grey-level morphological functionality [Adario, 99]. The real-time performance of DRIP was found to be 200% faster than execution with a comparative special purpose processor [Adario, 97]. Multi-FPGA solutions have also been found to deliver the highest performance levels within the image processing fields of region detection [Rachakonda, 95] and stereo vision [Vuillemin, 96].

The medical computing community have recognised the substantial performance capabilities of reconfigurable hardware [Martel, 00]. An element of this research has focused on combining the paradigms of medical image processing and reconfigurable computing. An example of this work is an implementation of a concentration index filter [Yokota, 02]. This filter calculates an index value for each pixel within a patient scan with the resultant output producing a diagnostic image. The filter has been applied in the detection of stomach cancer which occurs as gastric folds concentrate to form a cancer lesion. A reconfigurable hardware-based implementation of the filter operated a hundred times faster than a high-specification workstation and also facilitated real-time diagnosis.

Reconfigurable computing offers high performance and flexible solutions for cryptographic algorithms. A customised FPGA implementation of a cryptographic application is beneficial as the circuitry can be modified after production to correct security breaches or to ensure adherence to fresh security standards.

The PipeRench research project developed a pipelined reconfigurable fabric which virtualised hardware facilitating the execution of large cryptographic

[31] Dynamically Reconfigurable Image Processor

circuits on limited physical resources [Taylor, 99]. PipeRench is shown to outperform a high-performance general-purpose processor[32] by a factor of ten in the execution of the IDEA[33] cryptographic algorithm and it is also shown to be 40% faster than a full-custom silicon implementation of the algorithm. Reconfigurable hardware-based solutions have also been applied within the cryptographic domain to the MD5 hash algorithm [Deepakumara, 01] and to the RSA[34] algorithm [Vuillemin, 96].

Research groups specialising in multimedia-based systems have recognised the processing power and system flexibility achievable with reconfigurable computing solutions [Singh, 94]. A range of research has investigated opportunities for customising architectures for graphics applications. An example of this work has focused on executing geometric visualisation applications on a FPGA platform [Styles, 00]. Geometric visualisation involves the use of standard graphics rendering techniques such as texture mapping and uniform-direction lighting. Their results highlighted the FPGA platform is approaching the performance of a dedicated ASIC for general-purpose graphics applications. The reconfigurable hardware was 70% more efficient than a high-end workstation while achieving a 75% level of performance in comparison with the dedicated card. Additional research within the multimedia sphere has examined the potential of reconfigurable platforms to provide video decoding [Mignolet, 02] and speech recognition [Schmit, 95] services.

Reconfigurable computing systems can clearly deliver high performance solutions within a wide range of application areas. The computational traits of applications are important in determining their suitability for reconfigurable platform execution. The research above highlights the benefits that accrue when an appropriate application is implemented with reconfigurable technology.

[32] A 450MHz Pentium Workstation

[33] International Data Encryption Algorithm

[34] Rivest, Shamir, and Adelman

CHAPTER 4

Agent Middleware Empowering the Ubiquitous Computing Vision

4.1 Introduction

Agent technology is the middleware proposed for *pervasive handheld computing systems*. Four primary factors contributed to the decision to employ agent middleware to empower handheld devices to exploit reconfigurable resources within their environment.

Firstly, the agent-based paradigm is considered highly suitable for constructing modular software systems capable of operating in dynamic, unpredictable environments [Koch, 04]. Secondly, the paradigm is appropriate for analysing, specifying, and implementing complex software systems [Helin, 03]. Thirdly, agents can act as intelligent aids to users for advanced mobile services [Lino, 03]. Fourthly, agent middleware is widely considered as a primary enabling technology for empowering handheld devices within ubiquitous computing environments [Jennings, 98].

4.2 Agent Concept

In the context of software engineering, an agent can be defined as:

> An entity within a computer system environment that is capable of flexible, autonomous actions with the aim of complying with its design objectives [Wooldridge, 97].

An abstract view of an agent is presented in Figure 4.1. This diagram shows an agent taking sensory input from the environment and outputting actions which in turn affect the environment [Wooldridge, 02]. This interaction is usually ongoing and non-terminating.

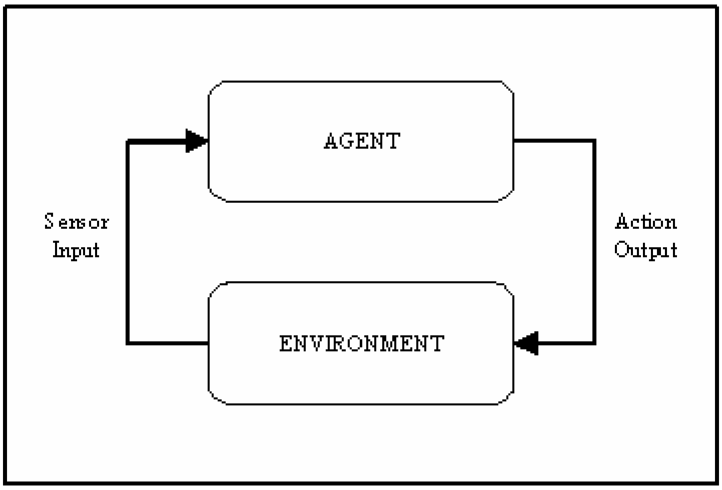


Figure 4.1: Generic Agent Model

An intelligent agent is built upon this generic agent model and has the following characteristics [Wooldridge, 95]:

- Reactivity
  Intelligent agents perceive their environment and can respond accordingly.

- Proactiveness
  Intelligent agents exhibit goal-directed behaviour and take the initiative whenever appropriate.

- Social Ability
  Intelligent agents are capable of interacting with other agents (and possibly humans).

FIPA[35] is a standards organisation whose aim is to promote the industry of these intelligent agents by openly developing specifications supporting interoperability among agents and agent-based systems [FIPA, 05a].

A FIPA-based model of the life cycle of an intelligent agent is shown in Figure 4.2 [FIPA, 05c].

[35] Foundation for Intelligent Physical Agents

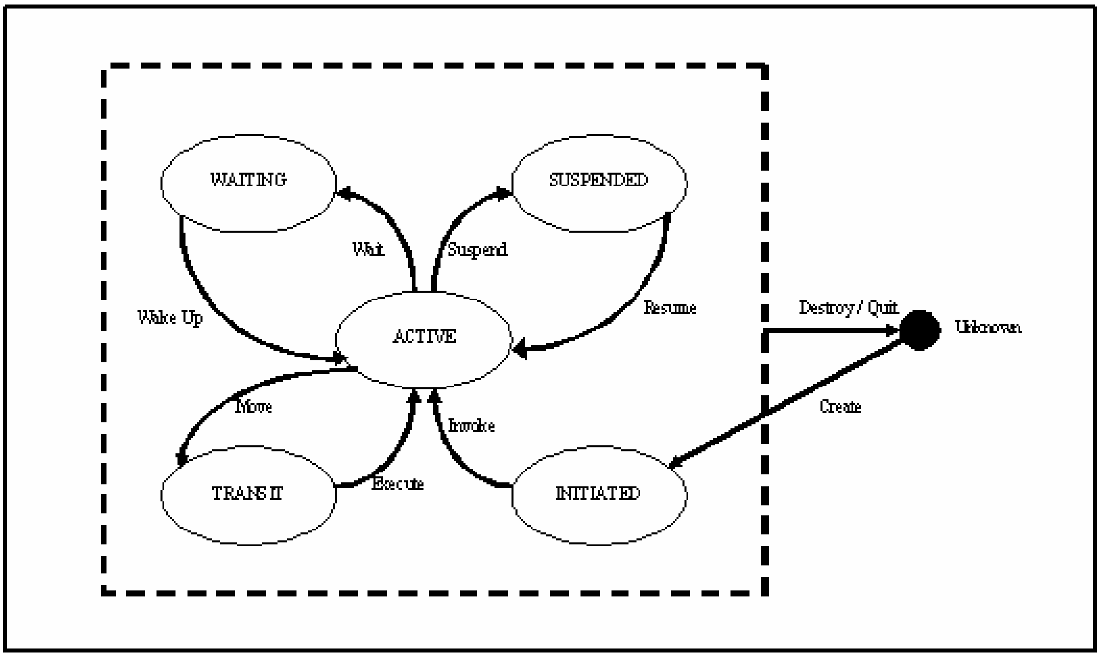


Figure 4.2: State Transition Diagram of Agent Life Cycle

This life-cycle model illustrates the state transitions that an agent can experience. The transit state is associated with the concept of a mobile agent.

4.3 Mobile Agent Concept

A mobile agent encapsulates the characteristics of an intelligent agent as well as having the added capability of traversing networks [Gray, 00]. The mobile agent concept is employed within the *collective past-experience learning strategy* outlined in Section 7.3.

The advantages of utilising mobile agents are outlined by Silva and Almeida in their examination of telecommunication systems [Silva, 99]:

- Network Traffic Reduction

  Mobile agents facilitate the concept of mobile code. The mobile code concept merges the models of *code-on-demand*[36] and *remote evaluation*[37] from the realm of distributing computing [Baldi, 97], [Fuggetta, 98]. Mobile agents transport code processing functions to the data source. This

[36] Java applets are an example of code-on-demand

[37] Java servlets are an example of remote evaluation

can conserve network bandwidth and reduce network interactions for applications processing large amounts of data.

- Software Upgrading
  Mobile agents can extend the capabilities of applications and can introduce new services in the machines or devices of the network. The *context-aware reconfiguration protocol* deployment strategy enhances this concept of software upgrading for handheld devices by incorporating dynamic configuration of reconfigurable hardware (ref. Section 6.3.3).

- High Robustness
  Mobile agents may have the ability to sense their execution environment and react autonomously to changes and failures in the network. This is in contrast to the client-server model which is highly dependent on network availability. This ability of mobile agents to cope with intermittent connections is highly beneficial within the domain of mobile computing [Kotz, 97], [Sahai, 98].

Mobile agent technology presents an alternative approach to the design of distributed systems as compared with traditional client-server and message-based architectures [Raibulet, 00].This alternative is not a universal solution and should be perceived as a complimentary technique to traditional object-oriented software development [Marques, 01].

4.4 Multi-Agent Systems

A multi–agent system is a federation of intelligent software agents interacting in a shared environment [Carabelea, 03]. A multi-agent environment encourages cooperation and coordination of actions amongst agents.

An agent management system is the software infrastructure used as an environment for executing multi-agent systems. The agent management system provides the normative framework within which agents exist and operate. Agent management establishes the logical reference model for the creation, registration, location, communication, migration and retirement of agents [FIPA, 05c]. The agent management reference model as specified by FIPA is shown in Figure 4.3.

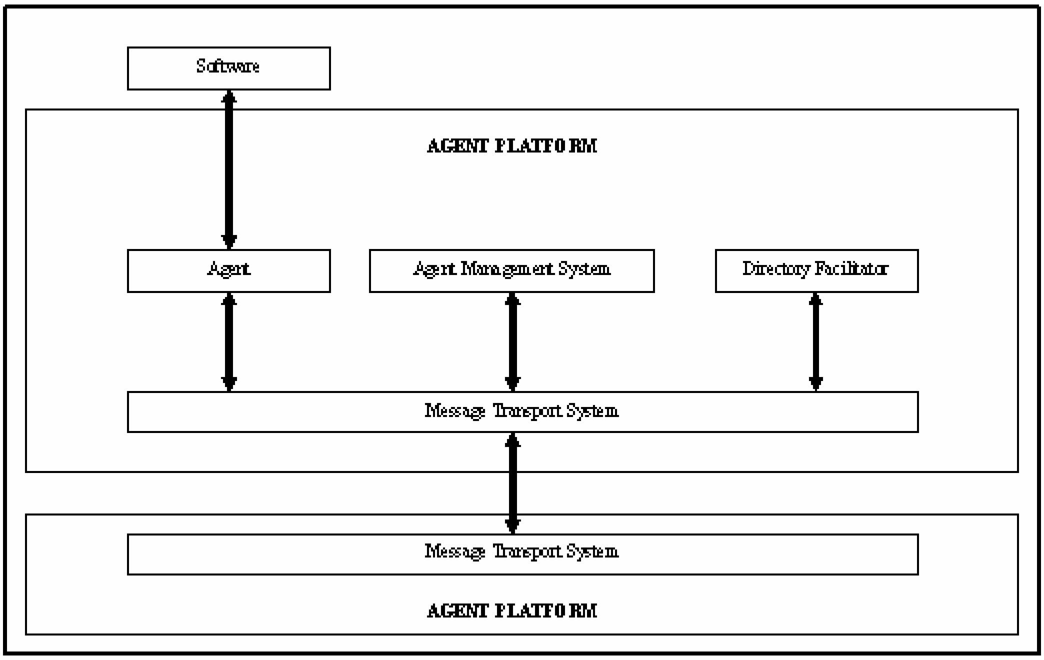


Figure 4.3: Agent Management Reference Model

Each of the logical components within the agent management reference model can be defined as follows:

- Agent
  This is a computational process that implements the autonomous, communicating functionality of an application. An agent is the fundamental actor on an agent platform which combines one or more service capabilities into a unified and integrated execution model.

- Agent Platform
  This is the physical infrastructure upon which agents are deployed. The agent platform consists of hardware, an operating system, and agent support software.

- Directory Facilitator

  This component provides yellow pages services to other agents. Agents utilise the directory facilitator to either register their services or query the services of other agents.

- Agent Management System

  This component exerts supervisory control over access to and use of the agent platform.

- Message Transport Service

  This component facilitates inter-communication between agents on different agent platforms.

Agents operating within an agent management system require an ACL[38] to communicate with other distributed entities. KQML[39] [Genesereth, 94] and FIPA-ACL[40] [FIPA, 05b] are the predominant ACLs utilised within the agent community. These communication languages are based upon speech act theory which was originally developed for modelling human communication [Helin, 03].

A common understanding of knowledge exchanged within a multi-agent system is facilitated through the use of a shared ontology [Jasper, 99]. An ontology consists of a vocabulary of terms and a specification of their meaning [Uschold, 98]. Essentially, an ontology is an explicit specification of a conceptualisation [Gruber, 93]. The ontology includes definitions and an indication of how concepts are inter-related. Collectively, this imposes a structure on the domain and constrains the possible interpretation of terms.

4.5 Agent Development Systems

The agent development systems employed to meet the goals of *pervasive handheld computing systems* are distinguishable according to their physical

[38] Agent Communication Language

[39] Knowledge Query and Manipulation Language

[40] Foundation for Intelligent Physical Agents - Agent Communication Language

platform dependencies. The physical constraints associated with portable computers have introduced a strand of research focusing on lightweight agent-based platforms. In contrast, provisioning and adaptive servers are capable of supporting more resource intensive agent platforms than handheld devices.

A wide range of agent development systems targeted towards server-based environments have been constructed. These systems provide predefined agent models and tools to ease agent-based development [Bellifemine, 00].

AgentBuilder provides two components named *toolkit* and *run-time system* for constructing Java-based agent systems [AgentBuilder, 00]. The *toolkit* consists of an integrated environment for managing the agent software development process. The *run-time system* is essentially an agent engine that provides an execution environment for the agent software.

dMARS is an agent-oriented development and implementation environment for building distributed systems based upon the BDI[41] agent model [D'Inverno, 97]. This type of agent model is associated with *deliberative* agent-based systems which encourages agents to reason about their actions. This development environment has been successfully applied to applications within the fields of air traffic control and business process management.

Zeus is an agent environment catering for rapid project development with a large library of agent components and an automatic agent code generator tool [Nwana, 98]. Zeus is primarily suited to the construction of *reactive* agent-based systems. Reactive systems operate in a stimulus-response fashion. The main role of reactive systems is to maintain an interaction with their environment, and therefore agents should be described and specified in terms of their on-going behaviour [Pnueli, 86].

[41] Belief, Desire, Intention

JADE[42] is a Java-based open source development framework aimed at developing multi-agent systems and applications [Bellifemine, 99]. The JADE project is the runtime environment employed within this thesis for provisioning and adaptive server-based prototype development and it is described in further detail in Section 4.5.1.

Agent development systems have also been targeted toward handheld device environments. This type of agent system can be classified into three categories [Ramparano, 02]:

- Portal Platforms
  This type of agent development environment does not support agent execution on the handheld device. The portable computer is only used as a visual interface to facilitate interaction with the mobile user. Agent operation and execution occurs on remote hosts. An example of this type of agent system is the MobiAgent development platform [Mahmoud, 01].

- Surrogate Platforms
  This type of agent platform supports partial agent execution on the handheld device. Parts of the agent execution model are run remotely on separate hosts. This *division of labour* amongst distributed entities produces light agents facilitating agent development for resource constraint devices e.g. mobile phones and two-way pagers. An example of this type of agent system is the kSACI development platform [Albuquerque, 01].

- Embedded Platforms
  Embedded platforms support the entire agent lifecycle and execution on handheld devices. This type of agent environment is utilised in the development of the medical-based prototype systems presented in Chapters Five, Six, and Seven. Examples of embedded platforms are AgentLight [Koch, 03], MicroFIPA-OS [Tarkoma, 02] and JADE-LEAP

[42] Java Agent Development Environment

[Caire, 02]. The JADE-LEAP[43] project is the runtime environment employed within this thesis for handheld device prototype development and it is described in further detail in Section 4.5.2.

4.5.1 JADE Agent Development System

JADE is a software framework for developing agent applications in compliance with FIPA specifications for interoperable intelligent multi-agent systems [Bellifemine, 03]. The JADE software architecture is fully developed in Java and is based upon the following driving principals:

- Interoperability
  JADE is compliant with all FIPA specifications. This ensures agents developed with JADE can interoperate with any FIPA-compliant agent framework.

- Uniformity and Portability
  JADE provides a homogeneous set of APIs that are independent of the underlying network and Java version support. Essentially, JADE maintains the same APIs for J2EE[44], J2SE[45] and J2ME[46] environments.

JADE enhances scalability by executing multiple parallel agent tasks within the same Java thread. This also helps meet the constraints of environments with limited resources. Additionally, JADE supports mobility of code and of execution state for J2SE and Personal Java environments. The mobility supported is of a *not-so-weak* nature because the stack and program counter cannot be saved in Java.

The JADE development environment represents a generalised agent model that can be specialised to realise both *reactive* and *deliberative* systems. The Jess[47]

[43] JADE Lightweight Extensible Agent Platform

[44] Java 2 Enterprise Edition

[45] Java 2 Standard Edition

[46] Java 2 Micro Edition

[47] Java Expert System Shell

software plug-in helps create *reactive* agent systems [Friedman-Hill, 03]. Jess is an expert system shell which fully integrates with JADE providing a reasoning engine for agents. Jess is outlined in further detail in Section 5.1.3.1. The Jadex[48] API is a rational agent layer that integrates with JADE facilitating the development of *deliberative* agent systems [Pokahr, 03].

4.5.1.1 Performance Evaluation of JADE

A number of performance evaluation surveys of agent development systems have been completed [Burbeck, 04], [Vitaglione, 02], and [Altmann, 01]. The JADE development environment has consistently been selected highly within these surveys in terms of maturity, security, communication facilities, memory efficiency, scalability, and performance.

Evaluation tests indicate that JADE is an efficient development environment limited mostly by the standard limitations of the Java programming language. The *write-once, run-everywhere* mantra of Java incurs an overhead in the form of its Virtual Machine. The JADE environment is found not to introduce substantial overhead [Vitaglione, 02].

For example, research has shown JADE has executed extremely efficiently on relatively antiquated hardware [Chmiel, 04]. Their evaluation environment comprised of PCs with Pentium II processors running at 120 MHz with 48 Mbytes of RAM and workstations with UltraSparc III processors running at 300 MHz and 192 Mbytes of RAM. This network environment supported experiments comprising of thousands of agents effectively migrating amongst the eight machines whilst also communicating tens of thousands of ACL messages. Their experiments also showed that an increase in the number of agents typically resulted in a linear increase of processing time.

Differences in execution time between the client-server and agent-based models are expected to diminish as the Java language matures and improves its efficiency

[48] Jade eXtension

[Chmiel, 04]. This interesting factor should be taken into account when evaluating the performance tests presented in Chapters 5, 6, and 7 of the thesis.

4.5.2 JADE-LEAP Agent Development System

JADE-LEAP is an agent-based runtime environment targeted towards resource constrained mobile devices [Berger, 03]. The main goal of the JADE-LEAP project is to develop a FIPA-compliant agent platform sufficiently lightweight to be deployed seamlessly on any Java-enabled handheld device [Bergenti, 01]. JADE-LEAP is an add-on to the JADE project replacing parts of its kernel creating a downsized agent platform [Moreno, 03].

The JADE-LEAP agent environment is operating system agnostic and can be executed on devices ranging from mobile phones and PDAs to workstations [Carabelea, 03b]. The agent platform is operational over any wireless network supportive of TCP/IP. The architecture of the platform is divided into several containers. A minimum of a single container is assigned to each physical computing element within the platform. Each of these containers may hold one or more agents. The containers are responsible for facilitating inter-communication amongst agents within the platform.

JADE-LEAP supports two separate forms of execution known as *stand-alone* mode and *split* mode. The former mode establishes a full container on the handheld device. The *split* mode divides a single container between a portable computer and a workstation. The container split creates a *front-end* container executing on the mobile terminal and a *back-end* container running within the fixed network. The *back-end* is delegated a large portion of container functionality allowing the *front-end* to become extremely lightweight in terms of memory and processing power consumption.

A *store-and-forward* mechanism is also implemented between the *front-end* and *back-end* containers. This deals with temporary wireless disconnections between the mobile terminal and the fixed network by buffering messages until wireless connectivity is re-established. The JADE-LEAP *split* mode supports extremely resource limited mobile devices and is the execution model chosen for the medical-based prototype systems presented within this thesis.

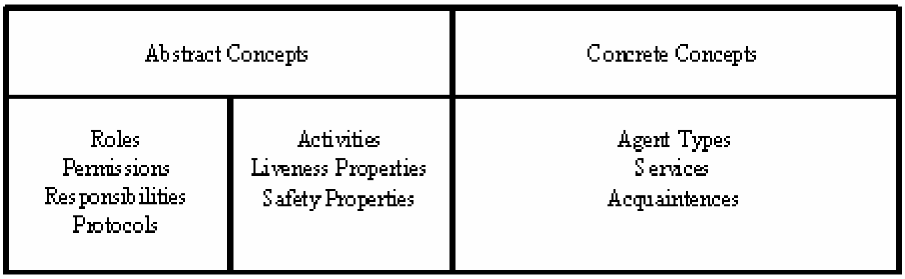
4.6 Agent-Oriented Analysis and Design Techniques

Software agent technology is a promising approach for the analysis, specification, and implementation of complex software systems [Helin, 03]. Agent-oriented analysis and design methodologies are essential components in creating agent-based solutions to complex systems.

This area of research is knows as AOSE[49] and is concerned with investigating agent-oriented design and analysis techniques. AOSE examines the engineering of software that has the concept of agents as its core computational abstraction [Weiss, 02].

4.6.1 Gaia Methodology

The Gaia[50] methodology is an agent-oriented analysis and design technique [Wooldridge, 99]. Gaia is employed in the conceptualisation and implementation of the medical-based prototype systems presented within this thesis. The main concepts of Gaia can be divided into the categories of *abstract* and *concrete* as shown in Figure 4.4 [Wooldridge, 02].



Abstract Concepts		Concrete Concepts
Roles Permissions Responsibilities Protocols	Activities Liveness Properties Safety Properties	Agent Types Services Acquaintences

Figure 4.4: Gaia Concepts

Abstract concepts are used during the analysis phase to conceptualise the system and its structure. The result of this process is captured in the *organisation* model. This model is a collection of roles, which stand in certain relationships to one

[49] Agent-Oriented Software Engineering

[50] The name originates from the Gaia hypothesis that all organisms in the Earth's biosphere can be viewed as acting together to regulate the Earth's environment.

another, and which take part in systematic, institutionalised patterns of interactions with other roles [Wooldridge, 99]. A role is defined by the following three properties:

- Responsibilities
  These determine the functionality associated with the role. Responsibilities can be divided into two types, namely, liveness properties and safety properties. Liveness properties describe a state of affairs an agent must establish given certain environmental conditions. In contrast, safety properties are invariants. A safety property ensures an acceptable state of affairs is maintained across all states of execution.

- Permissions
  A role is associated with a set of permissions. These permissions identify the resources available to the role to enable it to realise its responsibilities.

- Protocols
  A role is also identified with a number of protocols. These define interaction mechanisms with other roles.

The design aim of the Gaia methodology is to transform the models derived during the analysis stage into a sufficiently low level of abstraction that traditional design techniques (including object-oriented techniques) may be applied. The Gaia design process involves generating three models:

- Agent Model
  This documents the various agent types within the system. An agent type can be imagined as a set of agent roles.

- Services Model
  This identifies the main services associated with each agent role. A service is a function or a coherent block of activity of an agent.

- Acquaintance Model

  An acquaintance model defines the communication links between agent types.

The primary objective of the Gaia analysis and design process is to establish how a society of agents cooperates to realise the system-level goals. The methodology also helps to determine the requirements of each individual agent within this collaborating society.

4.6.2 Additional Methodologies

The predominant methodological approach within AOSE generally focuses upon adapting object-oriented analysis and design techniques. Examples of this adaptation strategy are the MaSE[51], AOAD[52], MASB[53], and AUML[54].

MaSE covers design and initial implementation using an agent modelling language and an agent definition language built upon OMT[55] and UML[56] [Wood, 00]. AOAD proposes an analysis and design method extending the concepts of class responsibility cards and responsibility-driven design from object-oriented development [Burmeister, 96]. MASB is an analysis and design method for agent-oriented systems which borrows models of behaviour diagrams, data models, transition diagrams, and object life cycles from object-oriented techniques [Moulin, 96].

AUML is the most well-known design technique at the forefront of this object-orientation adaptation strategy [Odell, 01]. It is an agent-oriented analysis and design process which builds upon both the Unified Modelling Language notation and the Rational Unified Process methodology [Booch, 98]. Agents are presented as an extension of active objects within AUML. Active objects exhibit the

[51] Multiagent Systems Engineering

[52] Agent-Oriented Analysis and Design

[53] Multi-Agent Scenario-Based

[54] Agent Unified Modelling Language

[55] Object Modelling Technique

[56] Unified Modelling Language

characteristics of dynamic and deterministic autonomy. These traits enable the object to initiate action without external invocation and to refuse external requests.

This active object to agent extension is indicative of the methodological approaches which modify and shape object-oriented analysis and design techniques towards agent-oriented requirements. The benefit of these strategies is considered to be the strength of employing UML software engineering concepts as a foundation for conceiving agent-oriented design and analysis methodologies.

The Gaia methodology outlined in Section 4.6.1 is in contrast to these object-oriented techniques and instead attempts to construct a framework which focuses upon the unique characteristics of multi-agent systems. This is a *purer* approach to agent-oriented development and it is the methodology employed within the medical-based prototype systems within this thesis.

Architects of the Gaia technique argue that a fundamental problem with methodologies that utilise object-orientation as a foundation is that they immediately fall short in their ability to capture the elementary characteristics of agency [Weiss, 02]. The differences between agents and objects are outlined in Section 4.7. This section presents the unique characteristics associated with agent-based computing in comparison with object-oriented development.

These differences between agents and objects clearly illustrate that an agent-oriented analysis and design strategy should be conceived primarily from agent-based computing concepts. This leads to an effective methodology enabling developers to achieve the correct decomposition of agent and object entities within their design.

4.6.3 Agent-based Design Patterns

The complexity of modern software and software environments has resulted in an increasing use of concepts and formalisms aimed at building applications more efficiently and cost-effectively [Weiss, 02]. Agent-based design patterns are an example formalism which improves the development process of applications and the quality of final products [de Araujo Lima, 03].

The development and usage of design patterns is becoming increasingly prevalent within the agent-based research community. Design patterns are a valuable mechanism enabling reuse of successful multi-agent structures [Gamma, 95]. Design patterns were originally conceived through the visionary work by Alexander in the field of architecture and urban planning [Alexander, 79]. In the context of software engineering, a design pattern can be defined [Buschmann, 96]:

> *As a particular recurring design problem that arises in specific design contexts and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.*

Essentially, patterns are reusable solutions to recurring design problems and they provide a vocabulary for communicating these solutions to others [Weiss, 04]. Aridor presents a catalogue of agent designs and representative patterns which are classified into three categories, namely [Aridor, 98]:

- Travelling Patterns
  These patterns deal with mobility management within agent-based computing systems. An *itinerary* pattern is an example pattern which is concerned with routing a mobile agent through multiple destinations.

- Task Patterns
  These patterns are concerned with the breakdown and delegation of tasks amongst agents. A fundamental task pattern within this classification is the *master-slave* design pattern. This entails a *master* agent delegating a task to a *slave* agent. The *slave* agent is instructed to move to a destination host, perform the assigned task, and return with an appraisal of task execution. This pattern is incorporated into the design of the *push-based configuration management deployment strategy* presented in Section 6.3.2.

- Interaction Patterns

  This classification of pattern is concerned with locating agents and facilitating their interaction. A *messenger* pattern is an example pattern which defines a surrogate agent to carry a remote message from one agent to another.

Further classifications of design patterns within agent-oriented computing include *agent patterns* which deal with the architectures of agent-based applications [Silva, 98], *communication patterns* which focus upon inter-communication between agents [Deugo, 99], and *coordination patterns* which are concerned with managing dependencies between agent activities [Tolksdorf, 98].

4.7 Differentiating Agents and Objects

Agents are defined as autonomous, problem-solving computational entities capable of effective operation within dynamic environments [Luck, 03]. Objects are defined as computational entities which encapsulate a state upon which they are able to perform actions [Wooldridge, 02].

The defining characteristic of object-oriented programming is the principle of encapsulation. This principle enables an object to exhibit autonomy over its state. However the principle does not extend to enable an object to exhibit control over its behaviour. Methods associated with an object are generally made available for other objects to invoke. Subsequently, an object has no control over whether one of its methods is invoked. This design methodology is favourable for a system composed of objects with a common goal. In contrast, a multi-agent system of communicating entities does not assume this common goal. Therefore, agents request actions to be performed rather than invoking methods directly.

Fundamentally, the locus of control with respect to decision-making regarding execution of actions differs between agent and object-oriented systems [Wooldridge, 02]. Object-oriented systems place the decision with the object invoking the method. Agent-oriented systems place the decision with the agent receiving the request. Essentially, agents can be distinguished from objects in that they are autonomous entities capable of exercising choice over their actions and interactions.

A second important distinction is the emphasis within agent-oriented systems upon flexible behaviour. The intelligent agent characteristics of reactivity, pro-activeness, and social interaction as outlined in Section 4.2 are integral to developing agent-oriented systems. Object-oriented design does not focus upon these characteristics.

Another important trait of agent technology is the promotion of a separation of concerns between computation, semantics, and interaction [Rimassa, 03]. An agent-oriented system deals with each of these aspects using the agent, ontology, and conversation protocol components respectively. This abstraction is integral within agent-oriented design whilst it is not apparent within object-oriented systems.

To summarise, the multi-agent paradigm is often compared to the distributed objects paradigm because both involve entities having an internal state, with an interface based on message-passing, and both allow for similar abstraction and modularity. However, the differences that make agents unique are:

(1) Their autonomy which means they maintain complete control over their actions whereas objects method invocation does not allow the same level of control.

(2) The emphasis placed within agent-oriented systems upon flexible behaviour.

(3) The promotion of a separation of concerns between computation, semantics, and interaction within agent-oriented systems.

Agent-based computing can be portrayed as extending object technology by enriching the component communication model and raising the abstraction level [Rimassa, 03]. However, leniency within the design of agent-oriented systems is also necessary. Just as structured control structures are apparent and useful within object-oriented systems; it may also be favourable to integrate passive components such as objects within agent-oriented systems. This is the basis of the mixed agent-object design technique presented in Section 4.8 which is conceived to improve the performance of agent-based systems.

4.8 Mixed Agent-Object Design Technique

The controversy of “agents versus objects” was a predominant focus within the agent-based research community. This debate challenged all aspects of agent-oriented analysis and design. An ideological shift within the community saw a movement away from this *either-or* strategy to recognition of a requirement for a *combined* approach. This change in perspective reflects a growing agreement on the need for both agents and objects [Weiss, 02].

The proposed mixed agent-object design technique advocates the use of both agents and objects within agent-oriented systems [O’ Sullivan, 06e]. The technique was initially conceived through an analysis of the performance results associated with the medical-based prototype systems presented within this thesis. Processing overhead of an *all-agent* based implementation for a recurrent computational task was considerably greater than the overhead required by a client-server based implementation. The disparity in execution times between the implementation models for the computational task is shown in Table 4.1.

Implementation Type	**Execution Time**
Client-Server Model	2711 ms
All-Agent Model	4075 ms

Table 4.1: Execution Times of Client-Server & All-Agent Models

This table highlights the processing time required by the all-agent model to be fifty percent longer than the execution time required by the client-server based model. The additional performance overhead can be primarily attributed to the multi-threaded nature of an agent-based implementation. This performance deficiency can be reduced and minimised by incorporating the mixed agent-object design technique within system development. The performance benefit of integrating this design approach is shown in Figure 4.5. This graph highlights the mixed agent-object design approach reducing processing overhead by 12% in comparison with the all-agent based implementation model.

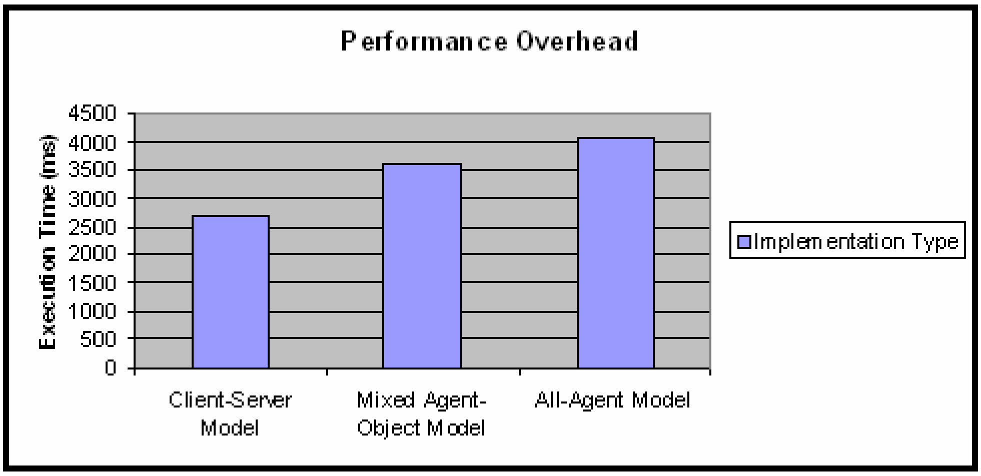


Figure 4.5: Performance Overhead of Implementation Models

The basis of the mixed agent-object design technique lies within a proposed modification to the Gaia methodology. This alteration strengthens the capability of the methodology to produce agent-oriented designs which achieve a good balance between agents and objects.

The mixed agent-object design technique is introduced within the analysis stage of the Gaia methodology during the creation of the roles model. This model identifies the key roles within the system. A role can be viewed as an abstract description of an entity's expected function. Currently, these roles are generally mapped to agent types during the design stage of the Gaia methodology. This leads to a danger within the agent-oriented development process of employing too many agents within the system design [Wooldridge, 98]. The performance results of the prototype systems clearly highlight the fact that the overhead of managing too many agent entities can rapidly outweigh the benefits of an agent-based solution.

The mixed agent-object design technique eliminates this danger within the Gaia methodology by achieving a fairer balance between agents and object types. The key opportunity to achieve this distribution occurs during role identification within the analysis phase. Essentially, the proposed modification advocates that the responsibilities associated with an identified role should exhibit two of the three key agent characteristics of reactivity, proactivity, or social ability for it to be mapped to an agent type during the design phase. A role not meeting this requirement should otherwise be mapped to an object type(s). This helps to minimise unnecessary agent instances within an agent-oriented design thus improving system performance. The mixed agent-object design technique is built upon a firm foundation as it is encapsulated within a widely accepted development methodology.

4.9 Agent-Based Solutions within Mobile Computing Environments

Agent-based computing is recognised as an enabling technology for next-generation mobile services [Jennings, 98]. Agent frameworks have formed an architectural basis for numerous applications within mobile computing environments. Service delivery has formed a focal point of development activity for agent-based researchers operating within the mobile computing field. The following taxonomy of agent-based mobile services has been presented [Koch, 04]:

- Granularity

  An agent-based mobile assistant system can be classified according to the granularity of the agent domain. A *single user support* system comprises of an agent framework which provides support to only one user. A *multi-user interaction support* system consists of an agent framework based upon a society of agents which cooperate to coordinate activities, negotiate resource usage, etc. The medical-based prototype systems presented within this thesis are within the category of *multi-user interaction support* systems.

- Role

  An agent-based mobile assistant system can also be classified according to the number of primary roles the system fulfils with respect to a categorisation of complex human-machine tasks.

  These tasks were initially presented by Sheridan [Sheridan, 98]. The medical-based prototype systems presented within this thesis fulfil varying levels of support for the following human-machine tasks:

  - Information Acquisition

    Agent system supports information gathering and storage.

  - Analysis and Display

    Agent system supports both analysis of collected information as well as graphical display of interpreted results.

  - Decision-Making Support

    Agent system supports intelligent analysis of collected information by evaluating data taking into account user context and preferences. The outcome of this sophisticated analysis procedure is a course of recommended action to the handheld device user.

  - Perform Action

    Agent system has the capability to carry out any actions required for the completion of a user objective.

- Autonomy

  An agent-based mobile assistant system can be categorised according to the level of autonomy exhibited within the application. The level of autonomy can be classified as either *reactive* or *proactive.* The actions of agents within a *reactive* system are triggered either by the user or the environment. Agents are capable of opportunistic goal-directed behaviour within *proactive* systems. The medical-based prototype systems presented within this thesis exhibit both *reactive* and *proactive* characteristics.

Real-world examples of agent-based mobile assistant systems include a software retrieval service [Mena, 00], [Mena, 02], a taxi management system [Moreno, 03], and a multimedia content provider service [Nikolouzou, 04]. The field of agent technology is also viewed as a highly suitable paradigm and inter-communication infrastructure for the analysis and design of mobile telemedicine systems [Della Mea, 01].

CHAPTER 5

Context-Aware Handheld Devices within a Medical Environment

5.1 Context Aware Computing

The goal of context-aware computing is to acquire and utilise information regarding the context of a device and to provide services that are appropriate to a particular setting [Bardram, 04].

Context-aware computing is a key component within two of the agent-based deployment strategies. The context-based negotiation strategy enabling utilisation of reconfigurable hardware as a networked resource depends heavily upon the contextual aspect of location to prioritise computational requests from medical practitioners. Additionally, the mobile device reconfiguration deployment scenario emphasising utilisation of reconfigurable hardware as an integral resource uses a range of contextual components to determine timing of configuration management operations.

Context-aware computing is a key element in providing the basis for intelligent handheld devices within these medical-based prototype systems. The precise benefits of utilising context within a healthcare environment can be explored more keenly by initially removing the aspect of reconfigurable hardware. This allows for an exact appraisal of the consequences of providing telemedicine applications on a medical practitioner’s mobile device with context-aware abilities.

5.1.1 CAMMD: Context Aware Mobile Medical Devices

CAMMD[57] is a framework conceived to examine the potential benefits of providing mobile medical devices with the ability to sense and interpret their contextual environment [O’ Sullivan, 06d]. CAMMD provides handheld medical devices with a support infrastructure capable of capturing, communicating and interpreting real-time contextual information. The framework focuses specifically

[57] Context Aware Mobile Medical Devices

on the proactive communication of patient records to a portable device based upon the active context of its medical practitioner.

Handheld medical devices can provide nomadic healthcare professionals with efficient access to patient records at the point of care. However, the storage and visual interface constraints of a portable device affects handheld analysis of these medical records. These delimiting factors combined with an intermittent wireless network connection can lead to an unsatisfactory user experience. The CAMMD framework investigates whether these issues can be resolved by allowing portable devices sense and interpret their contextual environment.

A context-aware mobile medical device can proactively assess its environment. The information gathered from this assessment can be interpreted to determine whether data management operations should be applied to the handheld device. This approach anticipates a medical practitioner's specific data requirements. Essentially, relevant patient records are proactively transmitted to a handheld device only when they are required.

The medical data to be propagated is determined using an informed decision-making process that evaluates the contextual environment of the handheld device. This methodology helps alleviate existing problems of information overload and low bandwidth. The proposed intelligent data management framework enhances the usability and portability of a handheld device. Additionally, the timely deployment of relevant medical records helps to eliminate handheld storage and visual interface constraints. These improvements can lead to increased productivity levels for medical practitioners and help to increase the accuracy of their patient diagnosis.

CAMMD utilises an agent-based architectural framework. Agent technology provides a sophisticated middleware capable of eloquently representing and communicating context-aware data elements. The nomadic nature of a medical practitioner emphasises location, time and activity as key context aware data components. These real-time data elements must be intelligently interpreted to inform the decision-making process within the agent framework. CAMMD utilises an expert system to determine whether data management operations are required for a handheld device. This rule-based system processes the raw ingredients of time and location of the handheld. These contextual elements are then cross-referenced with the work activity of the handheld user to determine whether data management operations are necessary. A pictorial representation illustrating the activities of this configuration management operation is shown in Figure 5.1.

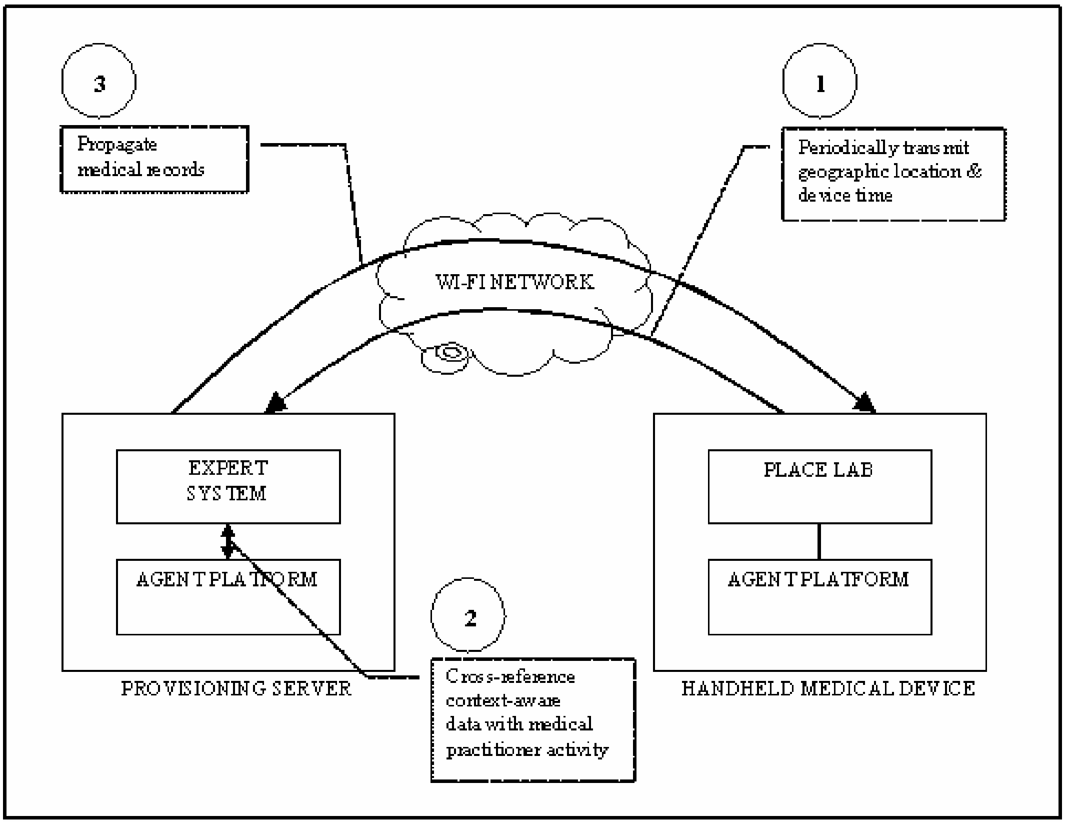

3
Propagate medical records
1
Periodically transmit geographic location & device time
WI-FI NETWORK
EXPERT SYSTEM
AGENT PLATFORM
PROVISIONING SERVER
2
Cross-reference context-aware data with medical practitioner activity
PLACE LAB
AGENT PLATFORM
HANDHELD MEDICAL DEVICE

Figure 5.1: Network Activity within CAMMD Configuration Management Operation

The contextual elements are examined by the expert system through firing a collection of pre-defined rules. Jess is the rule engine and scripting language employed within the framework. The Jess component and the technological role it plays within the proposed deployment scenarios for next-generation mobile medical devices is discussed in further detail in Section 5.1.3.1.

A primary contextual element required for successful deployment of the CAMMD is knowledge of the location of the handheld device. This is facilitated within the framework through the incorporation of a Place Lab module within each portable device. The Place Lab component and the technological role it plays within the proposed deployment scenarios for next-generation mobile medical devices is discussed in further detail in Section 6.2.2.2.1.

The contextual element of location has already been recognised as a useful mechanism to facilitate delivery of patient records to handheld devices [Rodriquez, 04a]. Their work recognises the importance of enabling intelligent handheld access to electronic medical records. This enhances device usability and improves the user experience.

CAMMD builds upon this work by placing an increased emphasis upon the need to intelligently interpret the contextual data elements of a handheld device. The expert system employed within the proposed methodology allows for a comprehensive analysis of these data elements. This enhances decision-making ability and enables the framework to deliver more appropriate and proactive support to users of handheld devices. A further consequence of this sophisticated support infrastructure is its ability to acutely manage physical device and network resources. Overall, CAMMD provides a framework to assess the implications of providing telemedicine applications on a medical practitioner’s mobile device with context-aware capabilities.

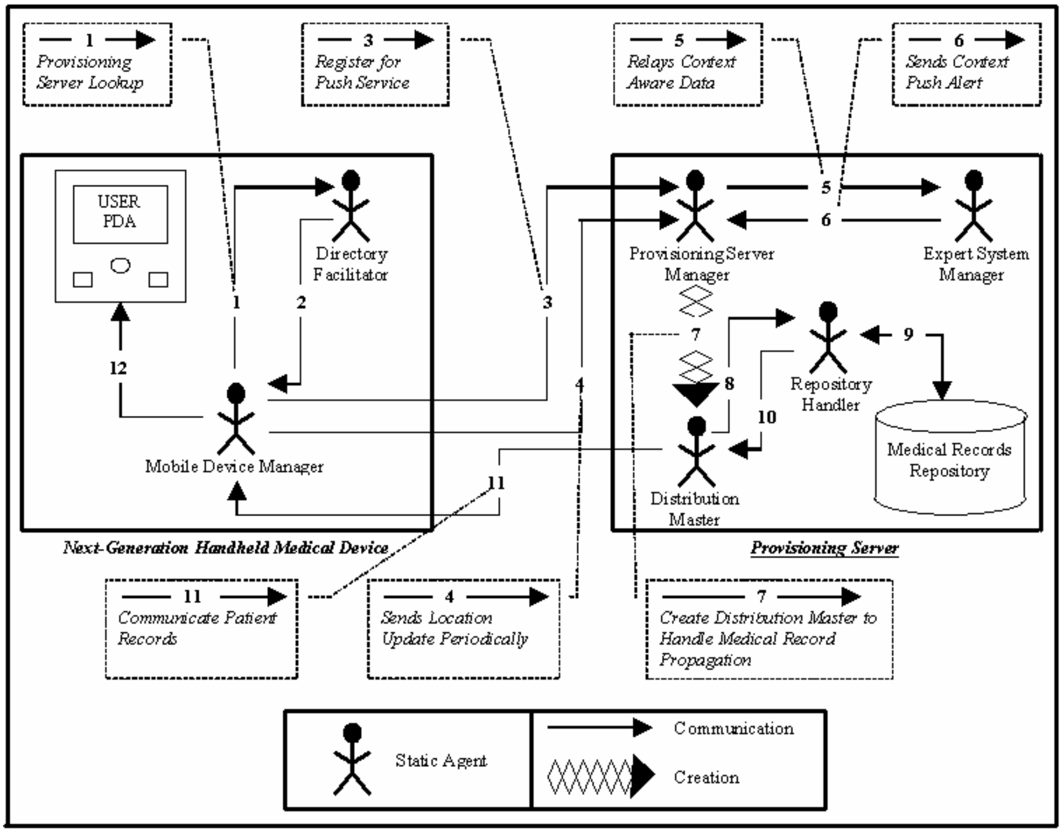

1 Provisioning Server Lookup
3 Register for Push Service
5 Relays Context Aware Data
6 Sends Context Push Alert
USER PDA
Directory Facilitator
Mobile Device Manager
Next-Generation Handheld Medical Device
Provisioning Server Manager
Expert System Manager
Repository Handler
Medical Records Repository
Distribution Master
Provisioning Server
1
2
3
4
5
6
7
8
9
10
11
12
11 Communicate Patient Records
4 Sends Location Update Periodically
7 Create Distribution Master to Handle Medical Record Propagation
Static Agent
Communication
Creation

Figure 5.2: CAMMD Agent Infrastructure

The CAMMD framework proactively communicates patient records to a portable device based upon the active context of a medical practitioner. Agent technology is the enabling middleware within this data management system. The agent infrastructure constructed to enable effective deployment of context-aware mobile medical devices is shown in Figure 5.2. This diagram highlights paths of intercommunication amongst agents as well as dynamic agent creation. The architecture was developed using an agent-oriented analysis and design methodology [Wooldridge, 99]. The role of each agent is outlined within Table 5.1. This table can be used as a reference guide to Figure 5.2.

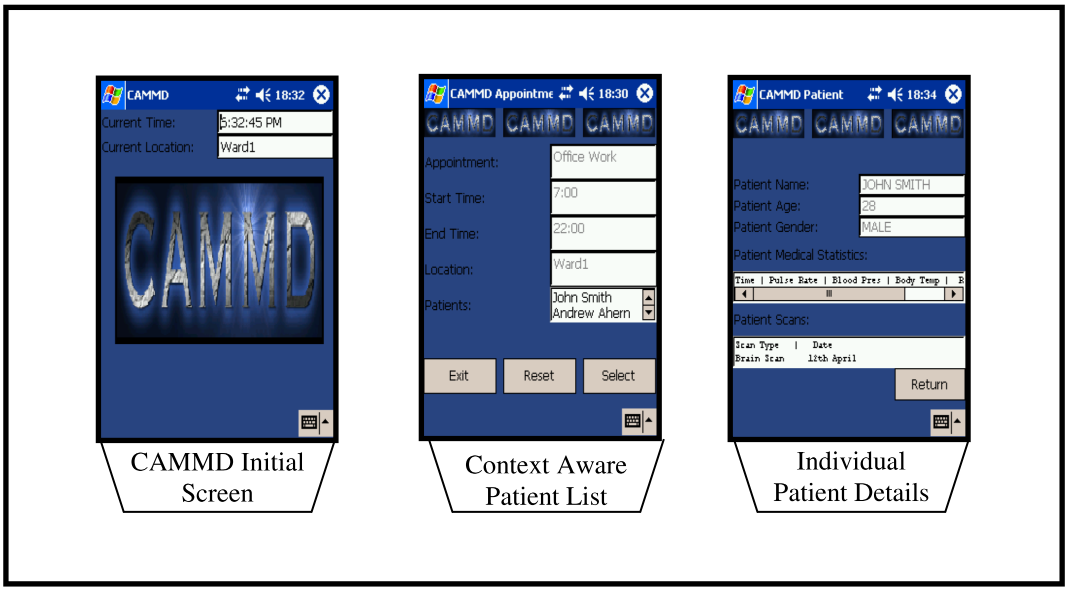
Agent Name	Agent Role
Mobile Device Manager	This single instance agent is a permanent resident on the mobile medical device and has responsibility for gathering and maintaining information about the physical device and its owner. The agent operates as the main point of contact between the user and medical applications. The agent registers for a medical record provisioning service. The operation of this service is based upon the contextual environment of the handheld device. The agent is also responsible for informing the provisioning server of any changes in the location of the handheld.
Directory Facilitator (DF)	The Directory Facilitator is responsible for maintaining knowledge about the location and services of each agent within the platform.
Distribution Master	This agent is instantiated as needed and is responsible for handling the propagation of patient records to a mobile medical device. This involves efficient inter-communication with the Repository Handler to obtain relevant records from persistent storage. These records are packaged into a medical-based message template and transmitted to the handheld device.
Provisioning Server Manager	This agent is responsible for the provisioning of electronic patient records to handheld medical devices based upon their active context. This agent accepts a request to provide a data management service to a portable device. The Provisioning Server Manager acts upon location updates from medical devices. These location alerts are triggered as the medical practitioner moves within the hospital. This information is communicated to the Expert System Manager to determine whether data configuration is required for the handheld device. A positive response from this agent will result in the creation of a Distribution Master agent to begin propagation of patient records to the mobile medical device.

Expert System Manager	The Expert System Manager maintains an interface to a rule-based expert system. This agent is responsible for controlling and interacting with the rule engine. This involves gathering the contextual data elements of a handheld device and communicating these values to the expert system. The decision of the rule engine informs the Expert System Manager whether data management operations are required.
Repository Handler	The Repository Handler interfaces with a medical database to obtain patient records.

Table 5.1: Agent Roles and Responsibilities within CAMMD Framework

5.1.2 Evaluation

An experimental prototype has been implemented to evaluate the performance of the CAMMD framework. This prototype facilitates the proactive communication of patient records to a portable device based upon the active context of its medical practitioner. Screenshots of this prototype in operation are shown in Figure 5.3.



Current Time: 5:32:45 PM
Current Location: Ward1

CAMMD Initial Screen

Appointment: Office Work
Start Time: 7:00
End Time: 22:00
Location: Ward1
Patients: John Smith
Andrew Ahern

Context Aware Patient List

Patient Name: JOHN SMITH
Patient Age: 28
Patient Gender: MALE
Patient Medical Statistics:
Time | Pulse Rate | Blood Pres | Body Temp | R
Patient Scans:
Scan Type | Date
Brain Scan 12th April

Individual Patient Details

Figure 5.3: CAMMD Prototype Screenshots

The left screenshot shows the graphical interface displayed to a medical practitioner upon initialisation of the CAMMD application. This screen displays the current time and location of the handheld device. The graphical interface is displaced upon receipt of a push of medical records from the provisioning server. This data management operation is triggered by the active context of the medical practitioner.

The propagated data consists of details related to a practitioner’s current appointment. This data transmission includes relevant patient records associated with the appointment. A visual representation of this propagated data as presented on the handheld device is shown in the middle screenshot of Figure 5.3. The graphical interface displays the location and time specific details related to the appointment as well as a list of associated patient names.

The rightmost screenshot shown in Figure 5.3 is generated upon the selection of a patient name from this list. The graphical interface displays the medical records of the selected patient. It includes general patient information and a list of their health diagnostics. The screen also informs the practitioner of any recent medical scans.

5.1.3 Test Case Environment

The test case environment consists of a Dell Axim PDA with a Pocket PC 2003 operating system. This handheld device executes the JADE-LEAP agent platform using a Personal Java virtual machine called Jeode. The provisioning server operates on a high-end Pentium PC running the JADE agent platform.

The Jess rule-based expert system resides on the provisioning server. Patient records propagated to handheld devices within the hospital scenario are stored in a SQL Server database. A Place Lab plug-in resides on each handheld device enabling an accurate location estimate to be communicated to the provisioning server. Agents communicate between the distributed components over a Wi-Fi[58] network.

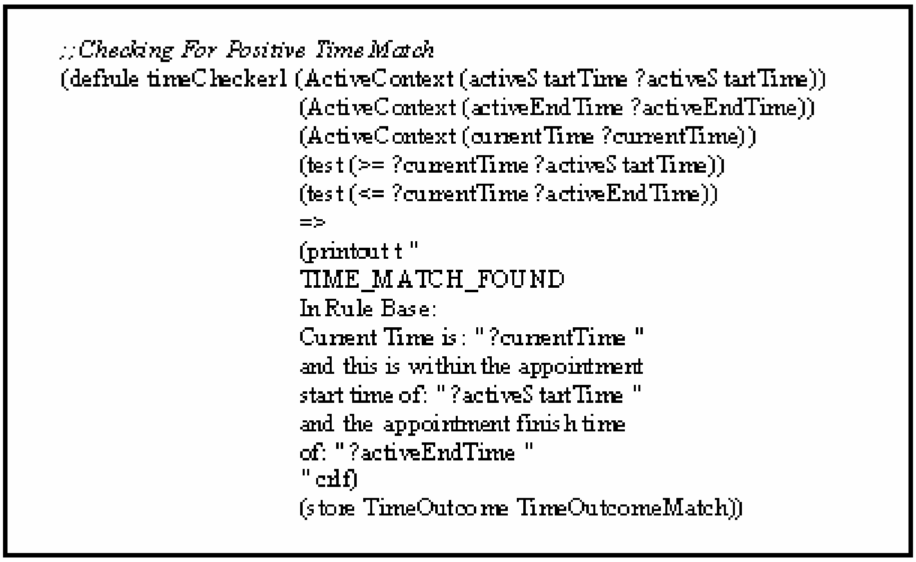
The test case deployment entailed assessing a CAMMD handheld device within a laboratory environment. The testing scenario attempted to closely emulate physical ward layout of Cork University Hospital.

5.1.3.1 Jess

Jess is an expert system shell which can be used as an agent reasoning engine [Friedman-Hill, 03]. This tool was inspired by the CLIPS[59] environment developed by NASA [Giarratano, 98]. Jess supports both forward and backward chaining and uses the Rete pattern-matching algorithm to process rules [Forgy, 82].

[58] Wireless Fidelity

[59] C Language Integrated Production System



```
;;Checking For Positive Time Match
(defrule timeChecker1 (ActiveContext (activeStartTime ?activeStartTime))
                      (ActiveContext (activeEndTime ?activeEndTime))
                      (ActiveContext (currentTime ?currentTime))
                      (test (>= ?currentTime ?activeStartTime))
                      (test (<= ?currentTime ?activeEndTime))
                      =>
                      (printout t "
                      TIME_MATCH_FOUND
                      In Rule Base:
                      Current Time is : " ?currentTime "
                      and this is within the appointment
                      start time of: " ?activeStartTime "
                      and the appointment finish time
                      of: " ?activeEndTime "
                      " crlf)
                      (store TimeOutcome TimeOutcomeMatch))
```

Figure 5.4: A Jess rule which cross-references appointment times with the current time

Jess is targeted towards Java-based platforms. The expert shell fully integrates with JADE agent development systems and helps to create *reactive* agent environments. Within the context of the CAMMD framework, Jess can interpret and evaluate the contextual elements of a portable device to recommend data management operations. An example rule within the CAMMD framework which cross-references the time aspect of a practitioner's schedule against the current time is shown in Figure 5.4.

5.1.4 Performance Results

Four individual tests were executed to evaluate the performance of CAMMD and these are outlined in Table 5.2. Each test was conducted using both the CAMMD framework and an RMI medical-based implementation. The tests operated within a simulated environment of ten geographically distributed wards. A timing scenario based upon guidelines for medical practitioner consultations was used as the test-case benchmark [BMA, 04].

Type	Test Name	Description
Physical Constraint Test	Handheld Device Storage	CAMMD Determine the storage cost on the handheld device resulting from the propagation of patient records. Remote Method Invocation Determine the storage cost on the handheld device resulting from a retrieval of patient records.
	Network Bandwidth Usage	CAMMD Determine the network bandwidth consumed by a CAMMD handheld device. Remote Method Invocation Determine the network bandwidth consumed by the RMI implementation.
Usability and Interaction Test	Data Transmission Time	CAMMD Determine the time taken to perform a data management operation. Remote Method Invocation Determine the time required for a retrieval of patient records from a provisioning server.
	User Navigation	CAMMD Determine the average user time to navigate to a patient medical record. Remote Method Invocation Determine the average user time to navigate to a patient medical record.

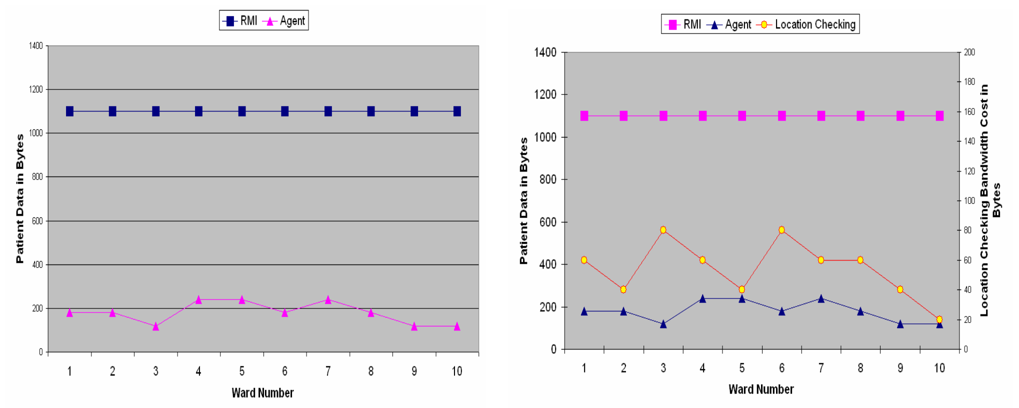
Table 5.2: Overview of CAMMD Performance Evaluation Tests

The British Medical Association report recommended a minimum of fifteen minutes per patient. The proposed use case scenario randomly distributed twenty-seven patients over ten wards to represent the daily workload of a medical practitioner. The patient to ward distribution is shown in Table 5.3. A walk-through of the wards was conducted by ten individuals to achieve results for each test case.

Ward Number	1	2	3	4	5	6	7	8	9	10
Number of Patients	3	2	4	3	2	4	3	3	2	1

Table 5.3: Patient to Ward Distribution

The first test examines the storage required by a CAMMD enabled handheld device when applying this use case scenario. Storage costs for the RMI implementation were also obtained. The results of this test case are shown in Figure 5.5.



Handheld Device Storage

Network Bandwidth Usage

Figure 5.5: CAMMD Physical Constraint Tests

The data storage on the PDA using the RMI implementation is constant due to the retrieval of every patient record for the medical practitioner at each ward. In comparison, the CAMMD implementation requires on average 80% less storage by retrieving patient records only associated with the practitioner's active context.

The second test examines the network bandwidth usage of a CAMMD enabled handheld device. Bandwidth usage of an RMI enabled device was also obtained. The results of this test case are shown in Figure 5.5. The network usage of the RMI enabled device is again constant and is calculated by determining the cost of invoking a remote retrieval of patient records. In comparison, the bandwidth usage of a CAMMD device fluctuates according to number of patient records transmitted and the frequency of location updates.

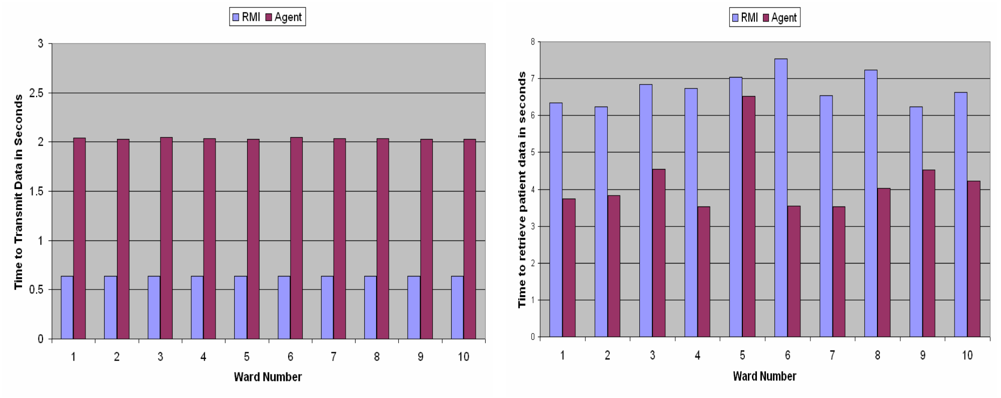
For example, test results for Ward 1 showed the bandwidth usage within the RMI implementation to be approximately 1100 bytes. The CAMMD test results for Ward 1 are based upon a series of location updates (right Y-axis) communicated to the provisioning server and the patient records (left Y-axis) propagated to the handheld device. The combined figures show a bandwidth usage of approximately 250 bytes highlighting an improvement of over 75% in relation to the RMI implementation[60].

The medical records are currently of a simple textual nature resulting in low memory requirements. Complex medical records with images of patient scans would show even greater disparity between RMI and CAMMD approaches in network bandwidth usage and handheld device storage requirements. The CAMMD framework clearly optimises the physical constraints of a handheld device and this improves device portability.

The third test examines the data transmission time of a CAMMD enabled handheld device. Transmission times of the RMI implementation were also obtained. The results of this test case are shown in Figure 5.6. Time to communicate patient records within the RMI and CAMMD-based prototypes is relatively constant. This is mainly due to the stability and availability of the wireless network. The results show the RMI implementation retrieves medical records on average three times faster than the CAMMD framework. The primary reason for this disparity is the inherent overhead associated with an agent

[60] Approximately 190 bytes required for patient records and 60 bytes required for location updates.

framework. The implications of this performance deficiency and a design mechanism to minimise its effects are discussed in detail in Section 4.8.



Data Transmission Time

User Navigation

Figure 5.6: CAMMD Usability / Interaction Tests

The fourth test evaluated the average time required by each user to navigate to a specific patient record in each ward. This test case examined the usability of both implementations. The results of this test case are shown in Figure 5.6. The concise nature of the patient records returned to a CAMMD enabled handheld device facilitated faster navigation times to individual patient records. The navigation time with the RMI-based implementation was on average two seconds slower. The primary cause of this delay is due to the extra time required to locate a specific patient within a larger list. The CAMMD implementation clearly improved user interaction by helping to avoid information overload.

5.1.5 CAMMD Appraisal

Healthcare organisations are increasing their reliance on mobile links to access patient medical records at the point of care. Mobile access to patient records improves the productivity of healthcare professionals and enhances the accuracy of their diagnosis. Handheld analysis of medical records is hindered due to the storage and visual interface constraints of a portable device. These physical constraints affect user interaction with handheld applications. This factor combined with an intermittent wireless connection can jeopardise the vision of a ubiquitous telemedicine environment.

The CAMMD environment typifies context-aware mobile medical devices. CAMMD provides a framework to assess the implications of providing telemedicine applications on a medical practitioner's mobile device with context-aware capabilities. The agent-based architectural solution proactively communicates patient records to a portable device based upon the active context of its medical practitioner.

This distribution of medical data enhances the usability and portability of mobile medical devices as shown in the usability and interaction test cases. The proposed methodology also overcomes handheld device and network limitation issues as shown in the physical constraint test cases. The CAMMD framework is a step towards realising the vision of a ubiquitous telemedicine environment.

CHAPTER 6

Agent Technology Exploiting Reconfigurable Resources for Handheld Devices

6.1 Introduction

The focus of this work is to investigate the potential of agent technology to exploit reconfigurable resources for handheld devices. The reconfigurable hardware is integrated into the environment of the portable computer, i.e. it is placed both into the physical device and into surrounding adaptive servers.

A number of deployment scenarios are proposed highlighting the benefits of agent technology as a middleware framework. These scenarios can be categorised into two groupings according to the placement of the reconfigurable hardware, namely:

- Reconfigurable hardware as a networked resource
- Reconfigurable hardware as an integral handheld device component

The proposed deployment scenarios within these groupings explore the benefits and consequences of empowering handheld devices with an agent-based middleware framework to exploit reconfigurable resources within their environment.

6.2 Reconfigurable Hardware as a Networked Resource

Adaptive servers are valuable distributed resources that can improve the system performance and versatility of client mobile devices. Reconfigurable hardware is the enabling technology of these adaptive servers. Reconfigurable resources within adaptive servers are costly commodities that should be effectively utilized to ensure satisfactory return on investment. They should also produce the highest possible performance and versatility gains for handheld devices. Agent technology is a highly appropriate middleware that can help meet this economic and system performance challenge.

Agents can be employed as an effective middleware to enable handheld devices assign their computational tasks to neighbouring adaptive servers. This offloading procedure entails intricate decision-making by agents operating both on the portable device and on neighbouring adaptive servers to ensure satisfactory performance. The construction and effectiveness of these decision-making frameworks benefits from the inherent characteristics of agents (ref. Section 4.2).

The following two deployment strategies outline the ability of agent middleware to empower handheld devices to exploit surrounding adaptive server technology:

- Agent-Based Negotiation Protocol
- Context-Based Negotiation Strategy

6.2.1 Agent-Based Negotiation Protocol

The proactive and cooperative characteristics of agents can be effectively integrated into a negotiation and bidding protocol enabling mobile devices gain access to networked reconfigurable resources [O' Sullivan, 05c], [O' Sullivan, 05d]. An agent-based negotiation protocol enables a handheld device efficiently offload reconfigurable hardware-software based computations to neighbouring adaptive servers.

This negotiation technique between handheld devices and adaptive servers for reconfigurable resources automatically introduces an effective load balancing strategy. A fair workload distribution is achieved amongst the adaptive servers. The distributed system is thus able to avoid both bottlenecks and under-utilisation of reconfigurable resources. This ensures a high quality of service to all handhelds through informed intelligent utilisation of reconfigurable resources.

The negotiation strategy between handheld devices and adaptive servers is embodied into an agent-based architectural framework as shown in Figure 6.1. This diagram highlights paths of intercommunication amongst agents as well as dynamic agent creation. The role of each agent is outlined within Table 6.1. This table can be used as a reference guide to Figure 6.1. The architecture was developed using the Gaia methodology [Wooldridge, 99]. An overview of this agent-oriented analysis and design methodology is presented in Section 4.6.1.

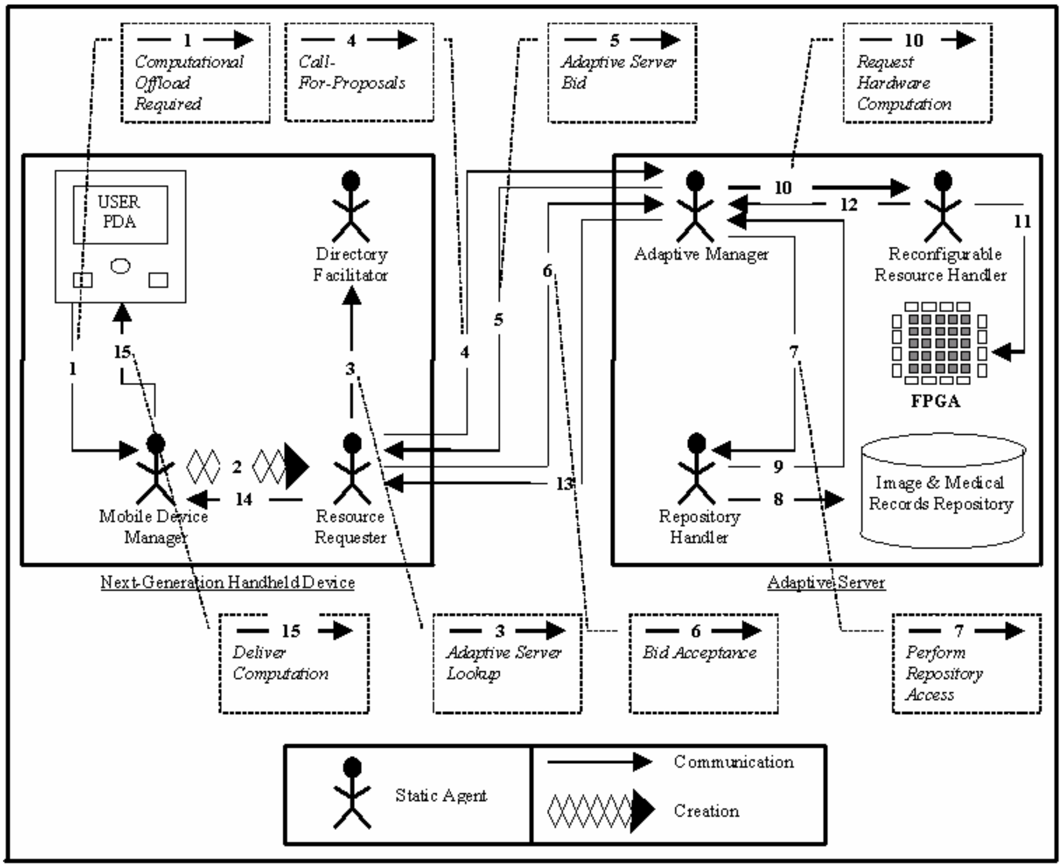


Figure 6.1: Agent Architecture Implementing Agent Negotiation Protocol

Agent Name	**Agent Role**
Mobile Device Manager	<u>*All Scenarios*</u> This single instance agent is a permanent resident on the mobile medical device and has responsibility for gathering and maintaining information about the physical device and its owner. The agent operates as the main point of contact between the user and the distributed adaptive server architecture. It predicts or responds to resource limitations on the mobile medical device by attempting to schedule a performance intensive computation upon a neighbouring adaptive server. The mobile device manager agent initialises the process of selection through invoking a resource requester agent. <u>Scenario 2: *Context-Based Negotiation Strategy*</u> The mobile device manager proactively determines the location of the portable computer. This contextual data element is communicated to a *Resource Requester* when starting the process of offloading a computation.
Directory Facilitator	<u>*All Scenarios*</u> This agent is responsible for maintaining knowledge about the location and services of each agent within the platform.
Resource Requester (Negotiator)	<u>*All Scenarios*</u> This agent is instantiated as needed and is responsible for initiating the negotiation process with adaptive servers for access to their reconfigurable resources. This negotiation strategy employs concepts based upon the contract-net protocol [Smith, 80]. The resource requester agent retrieves a list of all adaptive manager agents within the network from the directory facilitator. <u>Scenario 1: *Agent-Based Negotiation Protocol*</u> A *call-for-proposals* computation request is broadcast to all adaptive manager agents on this list. The resource requester then evaluates all adaptive manager bids to determine the best offload option. The computational task is assigned to the adaptive server which promises to service the request in the quickest time. This process of choosing an adaptive server helps maintain load balancing across all adaptive servers as it ensures fair

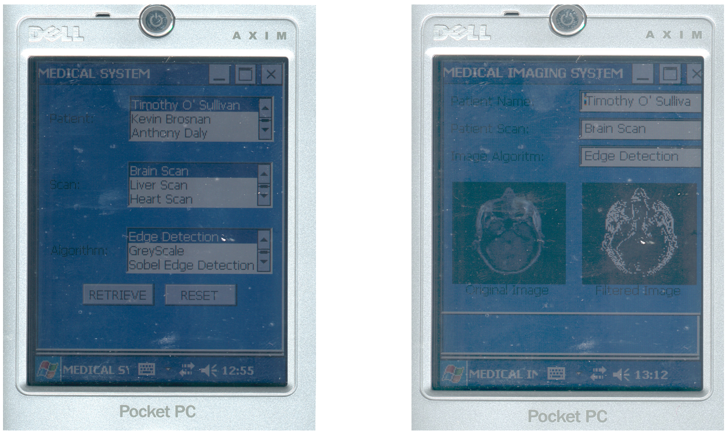
	workload distribution. <u>Scenario 2: *Context-Based Negotiation Strategy*</u> The agent performs all actions outlined in scenario one. The resource requester additionally populates a location field within the *call-for-proposals* computation request. This action communicates the originating location of the mobile device request to adaptive manager agents.
Adaptive Manager (Negotiator)	<u>*All Scenarios*</u> This agent is responsible for facilitating access to reconfigurable resources on an adaptive server. <u>Scenario 1: *Agent-Based Negotiation Protocol*</u> The adaptive manager plays a crucial role in establishing a load-balanced network by attempting to successfully bid to service a mobile medical device's computational request. Upon receiving a *call-for-proposals*, an adaptive manager examines its current queue of jobs and estimates their total service time. The result of this evaluation combined with an estimate of the time required to service the current computation request determines the adaptive manager bid. <u>Scenario 2: *Context-Based Negotiation Strategy*</u> The adaptive manager submits a bid to execute the handheld device computational request. The bid is determined by examining their current queue of jobs and estimating the total service time. This examination of the queue takes into account the geographic location of the current handheld device request. The priority level associated with the location of the incoming request dictates the placement of the potential computation within the adaptive manager's queue of jobs. The result of this evaluation combined with an estimate of the time required to service the current computation request determines the adaptive manager bid. The location aware aspect of the decision-making process identifies the urgency of the user request. A time-stamping operation is also employed by the protocol and this is applied to all tasks. This timing mechanism helps ensure lower level tasks avoid starvation by incrementing the priority level of each task periodically.
Repository Handler	<u>*All Scenarios*</u> This agent has responsibility for retrieving the reconfigurable bitstream representation of an algorithm required for a mobile device's computational request. The agent is also responsible for returning any additional data that may be required e.g. scanned patient images.

Reconfigurable Resource Handler	*All Scenarios* This agent is responsible for the process of downloading the bitstream configuration to the reconfigurable resource, interacting with the FPGA and communicating the results of the hardware computation to the adaptive manager.

Table 6.1: Agent Roles and Responsibilities within Networked Reconfigurable Hardware Deployment Strategies

6.2.1.1 Evaluation

A medical-based experimental prototype was constructed to evaluate the performance of the negotiation protocol. The prototype enables a physician to retrieve patient scans that have been image processed in real-time by an adaptive server. Screenshots of this prototype in operation are shown in Figure 6.2.


Patient: Timothy O' Sullivan, Kevin Brosnan, Anthony Daly

Scan: Brain Scan, Liver Scan, Heart Scan

Algorithm: Edge Detection, GreyScale, Sobel Edge Detection

Patient Name: Timothy O' Sulliva

Patient Scan: Brain Scan

Image Algoritm: Edge Detection

Figure 6.2: Medical Prototype Screenshots Implementing Agent-Based Negotiation Protocol

The left screenshot displays the options available to a physician in terms of patient names, associated scanned images and imaging algorithms that can be applied. The right screenshot shows a patient’s original brain scan and a filtered edge-detected image created in real-time by an adaptive server.

Edge detection algorithms are used widely in medical practise to aid physicians in their patient analysis. An edge detection algorithm implemented with reconfigurable hardware observes an increase in speed of a factor of twenty in comparison with an implementation of the algorithm in software [Daggu, 04]

6.2.1.2 Test Case Environment

The system architecture of the test case environment is presented in Figure 6.3.

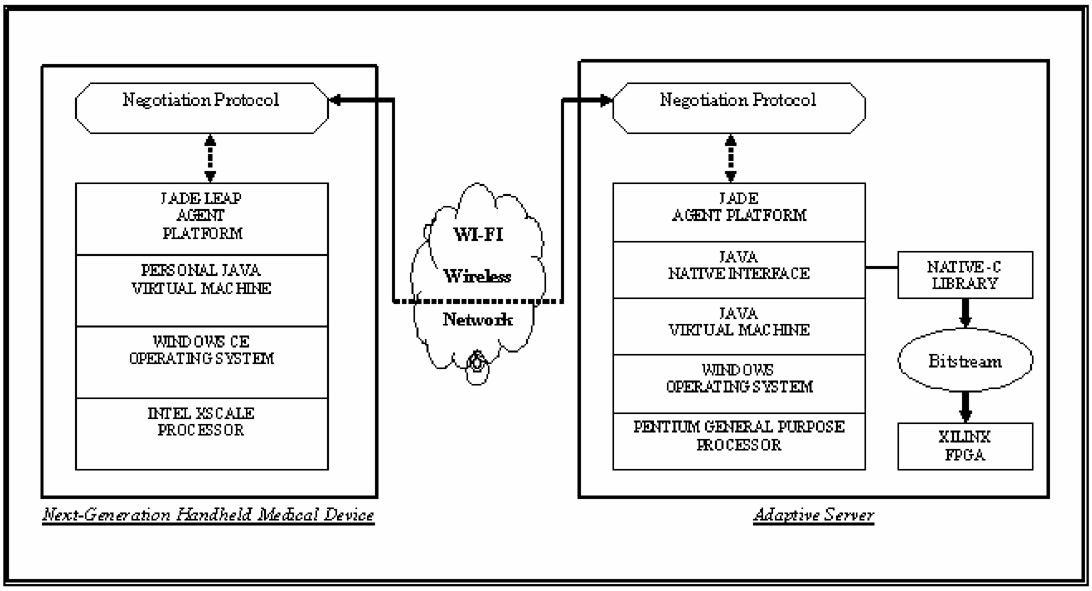


Figure 6.3: System Architecture for Agent-Based Negotiation Protocol

The handheld device component within the distributed medical framework consists of a Dell Axim PDA with a Pocket PC 2003 operating system [Dell, 05a]. The PDA executes the JADE-LEAP agent platform using a Personal Java virtual machine called Jeode [Jeode, 05].

Four adaptive servers execute within agent containers on a high-end Pentium PC executing the JADE agent platform. They are connected to a Celoxica RC200 reconfigurable hardware development board as shown in Figure 6.4 [Celoxica, 05a]. This development board is equipped with a Xilinx XC2V1000 FPGA [Xilinx, 05].

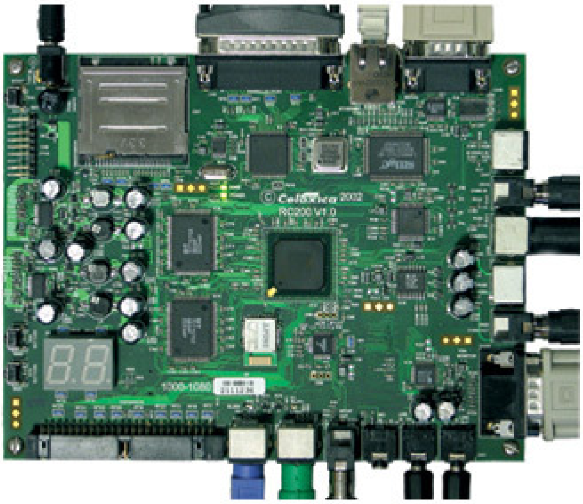


Figure 6.4: Celoxica RC200 Development Board

Agents communicate between the handheld device and the adaptive servers over a Wi-Fi 802.11b network [Belkin, 05]. The 802.11b specification supports bandwidth up to 11 Mbps[61] which is comparable to traditional Ethernet.

6.2.1.2.1 Agent Development Environment

JADE is a Java-based open source software development framework aimed at developing interoperable multi-agent systems and applications [Bellifemine, 99], [JADE, 05]. JADE is utilised as the active agent platform on all provisioning and adaptive servers.

The primary purpose of JADE is to simplify development while ensuring standard compliance through a comprehensive set of system services and agents. It can be considered an agent middleware that implements an agent platform and a development framework. JADE is completely implemented in Java with version 1.2 of the JAVA run time environment being the minimal system requirement [Java, 05].

[61] Megabits per second

JADE is an open source project being developed through a collaboration of Telecom Italia Lab (a research and development branch of the Telecom Italia Group) and the University of Parma.

JADE-LEAP is an agent-based runtime environment that is targeted towards resource-constrained mobile devices [JADE-LEAP, 05]. The JADE-LEAP module is an "add-on" to JADE replacing parts of its kernel creating a somewhat downsized agent platform. The JADE-LEAP runtime environment fully integrates with the JADE development infrastructure and it is the active agent platform on all handheld medical devices.

Both JADE and JADE-LEAP conform to FIPA standards for intelligent agents. FIPA is a standards organisation established to promote the development of agent technology [FIPA, 05]. JADE and JADE-LEAP are presented in further detail in Sections 4.5.1 and 4.5.2 respectively.

6.2.1.2.2 Reconfigurable Hardware Development Tools

The reconfigurable hardware development tools employed within prototype development of mobile medical computing devices are outlined within Section 3.6.

6.2.1.3 Performance Results

The effectiveness of the agent-based negotiation protocol was evaluated with the development of a purpose built simulator. A Java-based simulation environment was created and this used data obtained from the prototype system to achieve reliable analysis.

6.2.1.3.1 Simulation of Real-World Network States

Four separate scenarios were conceived to simulate various real-world network states. These network states are distinguishable according to the load balancing of tasks amongst the adaptive servers.

The scenarios model the potential variation of load amongst the four adaptive servers within the network. Pie chart representations of these probability distributions are shown in Figure 6.5. The probability distribution scenarios are utilised to populate adaptive server queues with pending tasks in an attempt to simulate various real-world network states. These task allocation scenarios can be described as follows:

- Equal Probability Distribution
  This scenario assigns an equal probability to each of the four adaptive servers of being allocated a job during the queue generation process. This distribution process will generate a state analogous to a load-balanced network.

- Low Variation Probability Distribution
  This scenario concerns a distribution modelling a network state with small variations of load amongst the adaptive servers. The variations are achieved by assigning slightly different job allocation probabilities to each server. These assigned probabilities are within a five percent range of the mean probability of twenty-five percent. This distribution process will generate a state analogous to a slightly unbalanced network.

- Medium Variation Probability Distribution
  This simulation scenario proposes job allocation probabilities between adaptive servers are within a range of fifteen percent of the mean probability. This distribution probability scenario will generate varying queues on each adaptive server resulting in a fairly unbalanced network state.

- High Variation Probability Distribution
  This probability distribution models high variations between each adaptive server load. The assigned probabilities are within a twenty-five percent range of the mean probability. This high variation creates large differences between adaptive server loads. The distribution process produces a network state analogous to a highly unbalanced network.

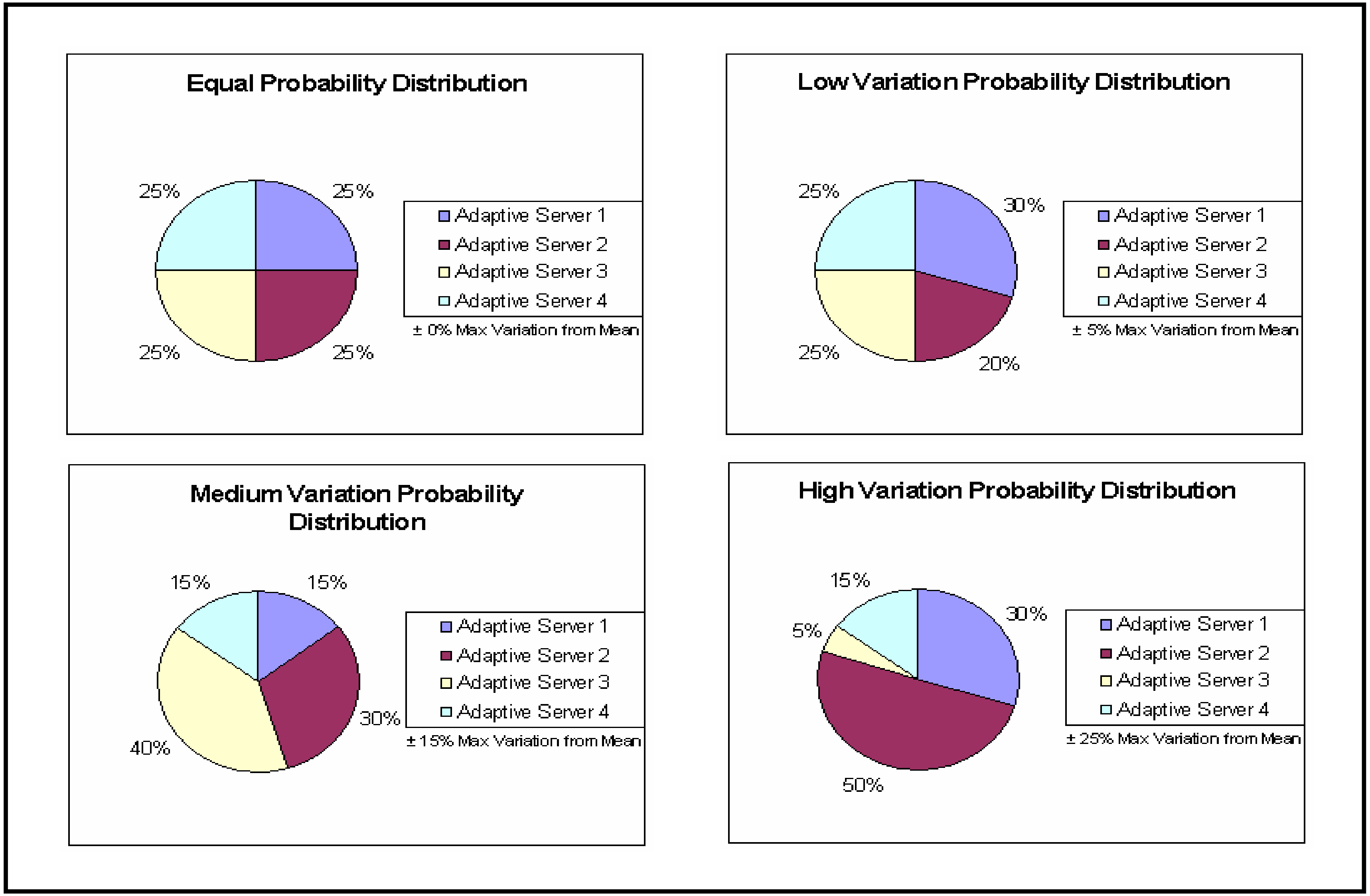


Figure 6.5: Load Distribution Scenarios Simulating Real-World Network States

The impact on queue generation by the four proposed probability distribution scenarios can be seen clearly in Figure 6.6. This graph highlights the job variance between adaptive servers for each simulated allocation.

All tests are executed in job increments of twenty-five using a purpose built Java simulator. The tests range from allocations of twenty-five jobs to five hundred jobs. The *Total Number of Jobs* in each test is allocated amongst the adaptive servers using the Monte Carlo random allocation technique [Metropolis, 49], [Rubinstein, 81].

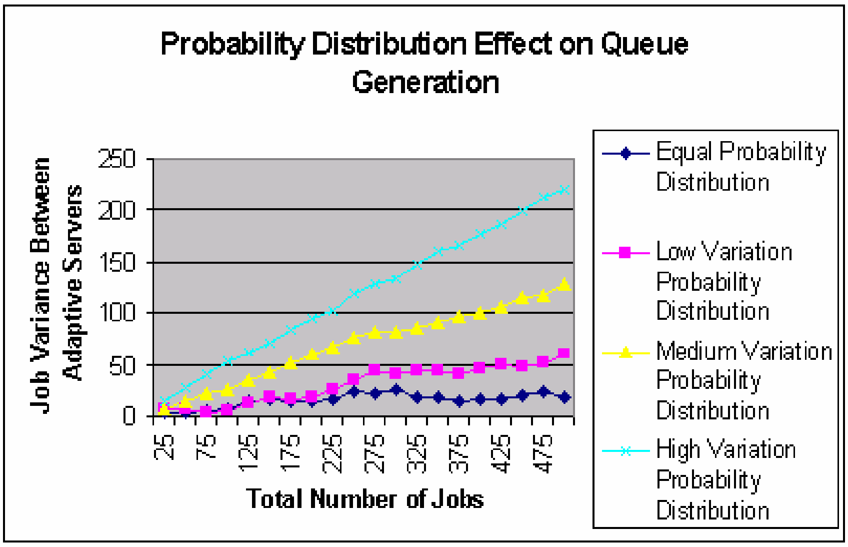


Figure 6.6: Effect of Probability Distribution Scenarios on Generation of Task Queues

The above graph highlights the variations in balance between the four adaptive server queues for each simulated probability distribution. The left axis displays the maximum differential between the jobs allocated to the adaptive servers at each test stage.

The balance produced by the *equal probability distribution* sees an initial rise followed by a relative stabilisation of the variance. There is 16% disparity between adaptive server computational loads initially but this averages out to 4.2% as job allocation figures increase. This illustrates that computational workload within this scenario would be shared comparatively equally amongst the four adaptive servers.

The simulation of the *low variation probability distribution* witnesses a gradual rise in the disparity of jobs assigned to adaptive servers. The variance is again high initially (i.e. 28% disparity) but overall has a mean of 11.8% for the *Total Number of Jobs* allocated amongst the adaptive servers. This probability distribution scenario represents a task allocation state analogous to a slightly unbalanced network.

The *medium variation probability distribution* observes a more disproportionate allocation of computational tasks amongst the adaptive servers. The job variance over all test stages averages at 26.7% for the *Total Number of Jobs* allocated. The queues simulated with this probability distribution are analogous to a fairly unbalanced network.

Finally, the simulation of the *high variation probability distribution* establishes a network state analogous to a highly unbalanced network. This probability distribution scenario produces widely disproportionate task queues for the adaptive servers. The job variance is approximately within a 44.2% to 60% ratio of the *Total Number of Jobs* allocated amongst the adaptive servers.

These probability distribution scenarios are representative of the varied load-balancing states of a networked environment. The effect of the distributions upon the queues of the adaptive servers illustrates network states ranging from a relatively load-balanced to a severely unbalanced infrastructure.

6.2.1.3.2 Time to Process Computational Request

A test was conceived to examine the time required to service a new computational request from a handheld medical device. The test compared the time expended by a device executing the agent-based negotiation strategy against the time needed by a device operating with a client-server based implementation.

This test examines the performance benefit delivered to a user of a handheld device equipped with an agent-based negotiation strategy. Any performance improvement could be effectively determined through a comparison with the client-server implementation.

The client-server implementation is an un-informed approach as it determines the adaptive server to be utilised through a random selection of one of the available servers. The negotiation strategy is an informed approach which facilitates the choice of adaptive server with the lowest computation load (i.e. smallest job queue) at runtime.

The simulation employed quantitive values from test-case analysis of both the agent-based and simple client-server based implementations. The time to process an edge detection computation request from a Wi-Fi PDA on an adaptive server using the agent-based negotiation technique was determined. This value comprises of the intercommunication time required for negotiation and also the time for the adaptive server to process a computational request. The execution of a computational request is the time required for the adaptive server to:

- Initialise the FPGA
- Download the configuration bitstream
- Pass the image data
- Execute the algorithm through clocking the FPGA
- Read back results from the FPGA.

The intercommunication and execution time of an edge detection computational request from a Wi-Fi PDA to an adaptive server using a simple client-server implementation was also determined. This time was established using a Java-based remote method invocation (RMI) implementation [Java, 05b].

A breakdown of these quantitive time values is shown in Table 6.2. These values were determined through an averaging of results achieved from twenty separate test-runs of both experimental prototype implementations.

Implementation Type	**Intercommunication Time**	**Adaptive Server Execution Time**
Agent-based Negotiation Model	3110 ms[62]	4075 ms
Client-Server Model	1781 ms	2711 ms

Table 6.2: Average Quantitive Times for Agent-Based Negotiation

The execution time of a computational request by an adaptive server implementing the agent-based negotiation protocol is 50% longer than the equivalent execution within the client-server based model. Additional performance overhead can be primarily attributed to the multi-threaded nature of an agent-based implementation. The implications of this performance deficiency and a design mechanism to minimise its effects are discussed in detail in Section 4.8.

The intercommunication time of the agent-based model is also over 70% longer than the client-server model. This is primarily due to the bidding aspect of the agent-based model which demands a number of interactions between the handheld and server platforms. These additional interactions are crucial as they inform the decision-making of the *ResourceRequester* agent operating on the handheld device. The benefit of this knowledge to the user of a mobile device can be seen in the graphs presented in Figure 6.7.

These graphs show the time to process an offloaded computational request from a Wi-Fi enabled handheld device using the agent-based negotiation model and the client-server based model within the simulated network environments. These graphs are generated with results from a purpose-built Java simulator using the quantitive values presented in Table 6.2. The jobs allocated to the adaptive

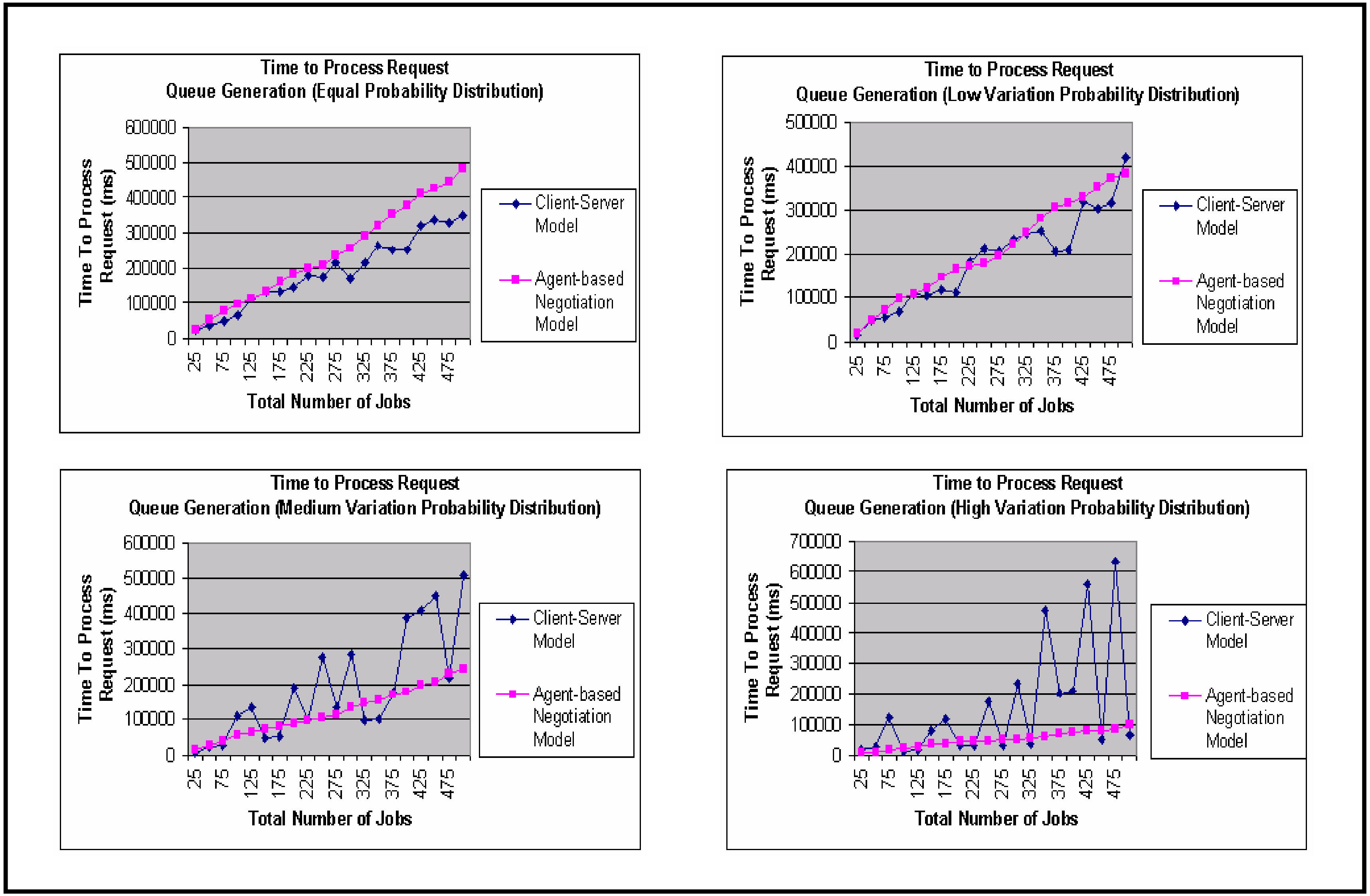
[62] Millisecond

servers within the simulation are derived from the probability distribution scenarios shown in Figure 6.5.

The agent-based negotiation protocol ensures the handheld device always offloads its computation to the adaptive server with the smallest queue of jobs. The client-server approach chooses the adaptive server to offload its computational request through a random selection of one of the available adaptive servers. This uninformed decision-making means the client-server model generally only delivers quicker service to the user of a handheld device when the randomly selected adaptive server happens to have the smallest load. The likelihood of this occurrence proportionally decreases as the number of adaptive servers within the network environment increases.

Each probability distribution graph within Figure 6.7 highlights the time required to service a new computational request utilising the client-server and agent-based models. The unpredictable nature of the client-server model in relation to expected service time is highlighted as the network environment becomes increasingly unbalanced. These wide ranging times lead to poor quality of service for the user of a handheld device. The wide disparity between expected service times is also indicative of poor utilisation of network resources.

In contrast, the agent-based negotiation protocol encourages optimal utilisation of networked reconfigurable resources through informed decision-making for offloading computation. There is a linear relationship between the service time for a handheld computational request and the total number of jobs awaiting process within the distributed environment. The gradient of this linear relationship is dependent on the equality of task distribution across the adaptive servers. The protocol enables a mobile device to accurately estimate the service time for an offloaded task helping to enhance the quality of service for a handheld user.



Time to Process Request
Queue Generation (Equal Probability Distribution)

Time to Process Request
Queue Generation (Low Variation Probability Distribution)

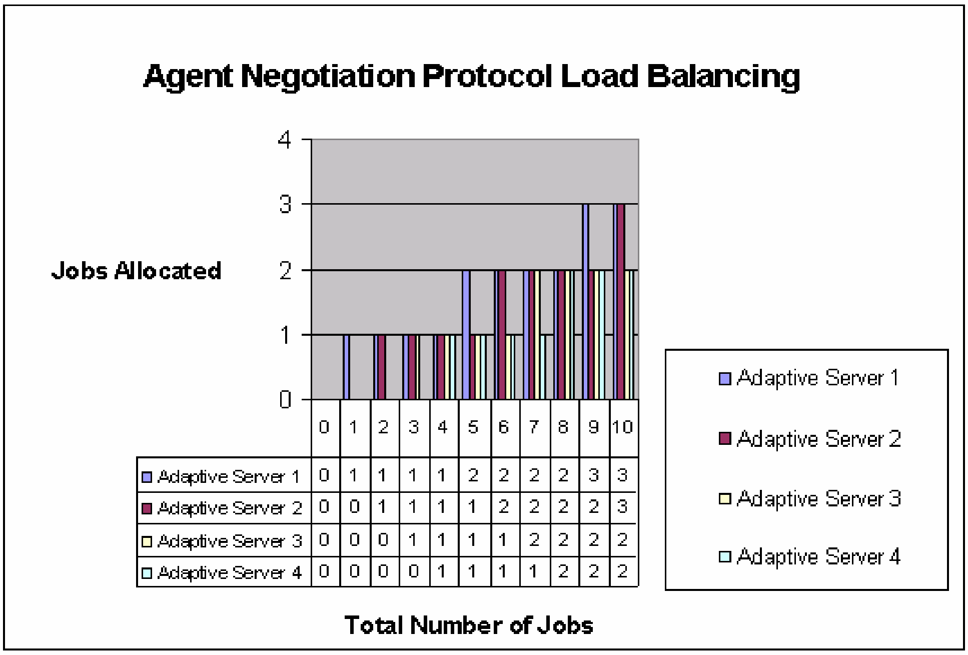
Time to Process Request
Queue Generation (Medium Variation Probability Distribution)

Time to Process Request
Queue Generation (High Variation Probability Distribution)

Figure 6.7: Time to Service a Computational Request

6.2.1.4 Load-Balancing Effect of Agent-Based Negotiation Protocol

The test simulation examining the time to service a new computational request from a handheld device also clearly shows the protocol attempts to re-establish load balancing across the adaptive servers regardless of current network state. A simulated load-balancing graph illustrating this aspect is shown in Figure 6.8.



	0	1	2	3	4	5	6	7	8	9	10
□ Adaptive Server 1	0	1	1	1	1	2	2	2	2	3	3
■ Adaptive Server 2	0	0	1	1	1	1	2	2	2	2	3
□ Adaptive Server 3	0	0	0	1	1	1	1	2	2	2	2
□ Adaptive Server 4	0	0	0	0	1	1	1	1	2	2	2

Figure 6.8: Agent Negotiation Protocol Load Balancing Scenario

The simulation assumes all tasks are offloaded using the agent-based negotiation protocol and presupposes a reliable and robust network. The graph shows the job variance between adaptive servers for each simulated allocation within an ideal network environment to be no greater than one. This highlights the ability of the agent-based negotiation protocol to lend itself greatly to establishing a load-balanced network.

6.2.2 Context-Based Negotiation Strategy

A context-based negotiation strategy within an agent-based framework enables intelligent utilisation of surrounding reconfigurable resources by mobile devices [O' Sullivan, 05a]. A context-aware handheld can proactively assess its environment. The information gathered from this assessment can better inform the decision-making process of agents operating within adaptive servers with regard to resource allocation.

The execution of the offloading protocol is influenced by the location of the portable device. This contextual information enables an adaptive server to identify the urgency of a computational request from a handheld. The identification of task priority based upon the location of the mobile device is reflected in the adaptive server response to the computation request. This helps to optimise the quality of service experienced by a handheld device user.

An example deployment of relationships between location and task priority within a telemedicine environment is shown in Table 6.3.

Handheld Device Location	**Priority Level**
Emergency Room	Urgent
Hospital Ward	High
Hospital Corridor	Medium
Practitioner Office	Low

Table 6.3: Example Priority Levels within Telemedicine Environment

This table presents an association between the geographic location of a medical practitioner and the priority level assigned to their handheld device computational requests. This priority level reflects the urgency of their offloaded tasks.

The agent architecture developed to facilitate this context-based negotiation strategy is shown in Figure 6.9. This diagram highlights paths of intercommunication amongst agents as well as dynamic agent creation. The architecture was developed using an agent-oriented analysis and design methodology [Wooldridge, 99]. The role of each agent is outlined within Table 6.1. This table can be used as a reference guide to Figure 6.9.

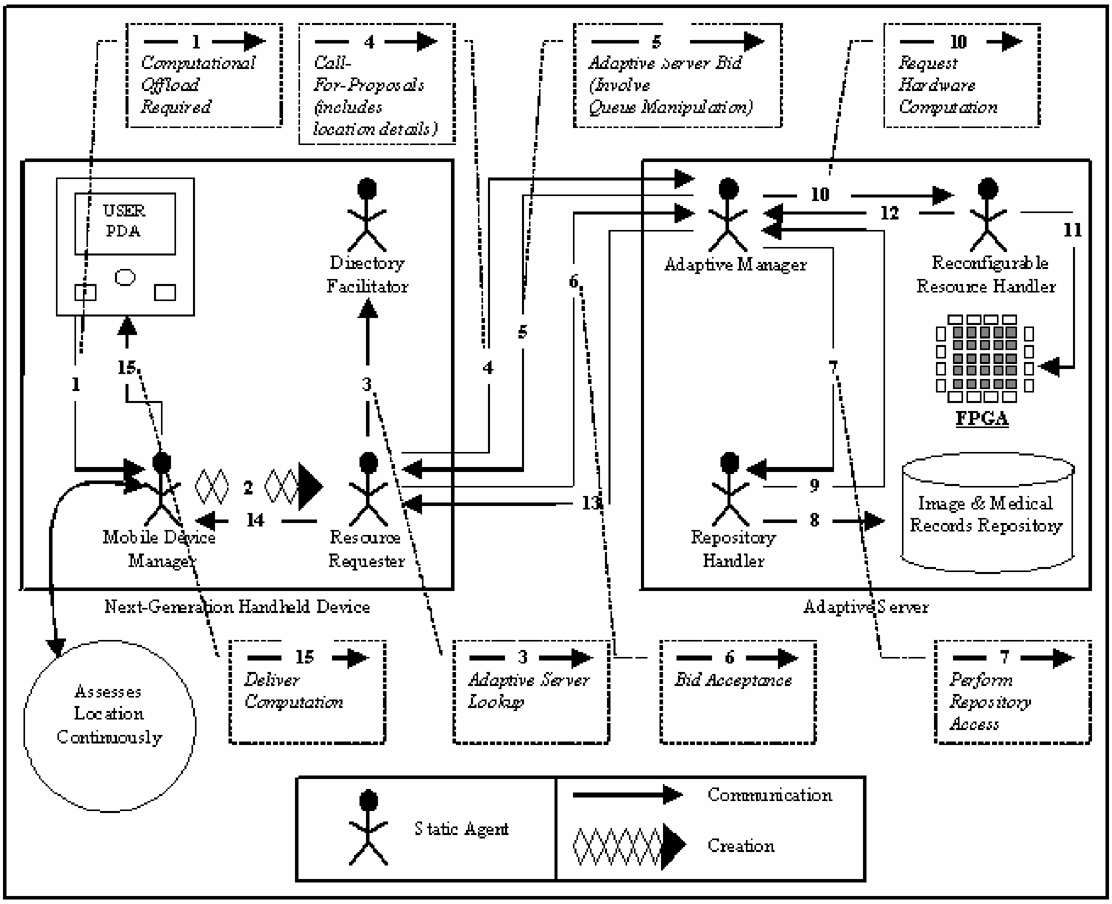
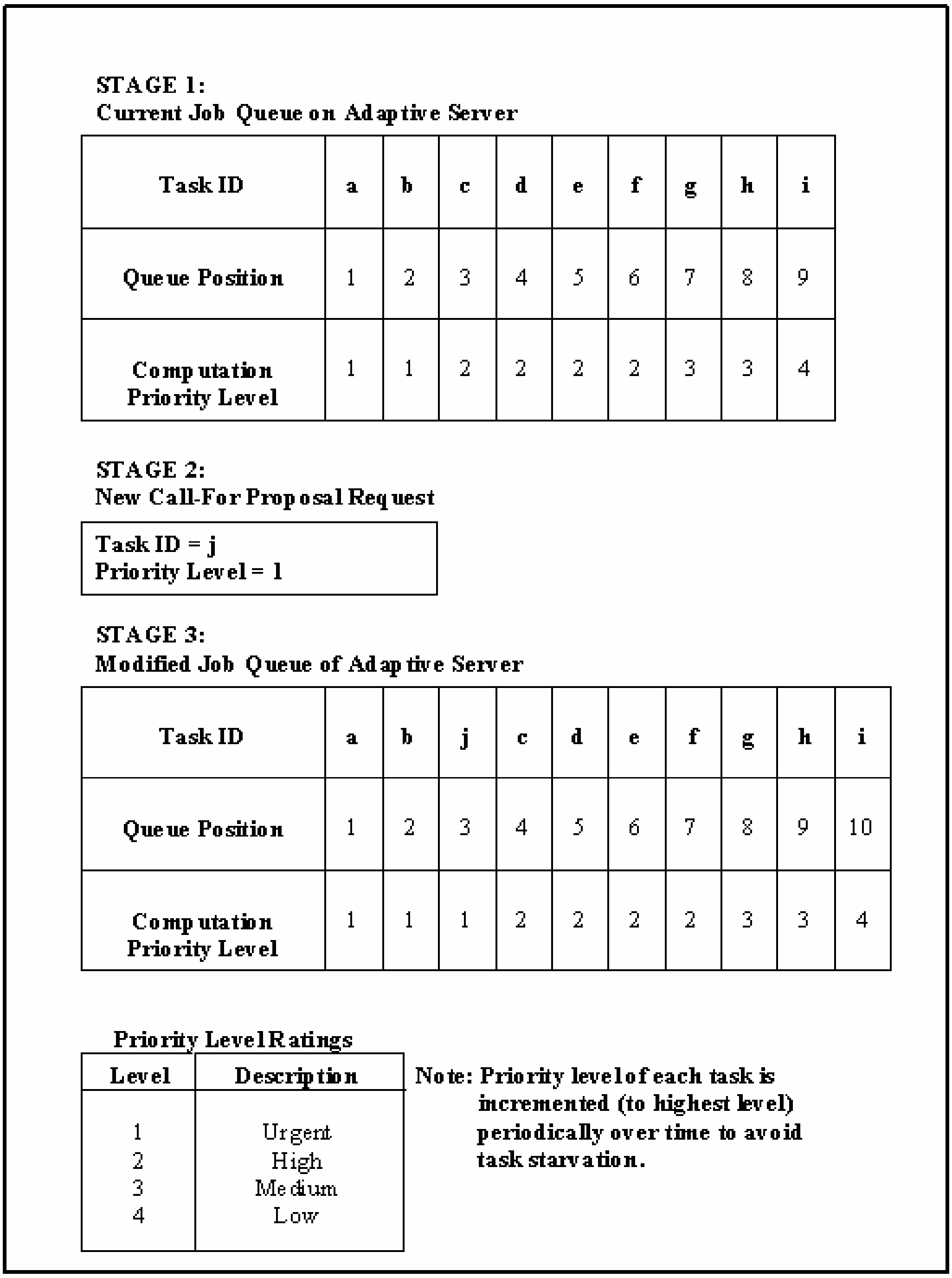

1 Computational Offload Required
4 Call-For-Proposals (includes location details)
5 Adaptive Server Bid (Involve Queue Manipulation)
10 Request Hardware Computation
USER PDA
Directory Facilitator
Adaptive Manager
Reconfigurable Resource Handler
FPGA
Image & Medical Records Repository
Repository Handler
Mobile Device Manager
Resource Requester
Next-Generation Handheld Device
Adaptive Server
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Assesses Location Continuously
15 Deliver Computation
3 Adaptive Server Lookup
6 Bid Acceptance
7 Perform Repository Access
Static Agent
Communication
Creation

Figure 6.9: Agent Architecture Implementing Context-Based Negotiation Protocol

A key aspect within the context-based negotiation strategy is the manipulation of the adaptive server queue of tasks by the *Adaptive Manager* agent. This modification of task order within the queue is dependent upon the priority level of the incoming computation request. An example scenario of modifications to a queue of jobs by an *Adaptive Manager* is shown in Figure 6.10.



STAGE 1:
Current Job Queue on Adaptive Server

Task ID	a	b	c	d	e	f	g	h	i
Queue Position	1	2	3	4	5	6	7	8	9
Computation Priority Level	1	1	2	2	2	2	3	3	4

STAGE 2:
New Call-For Proposal Request

Task ID = j
Priority Level = 1

STAGE 3:
Modified Job Queue of Adaptive Server

Task ID	a	b	j	c	d	e	f	g	h	i
Queue Position	1	2	3	4	5	6	7	8	9	10
Computation Priority Level	1	1	1	2	2	2	2	3	3	4

Priority Level Ratings

Level	Description
1	Urgent
2	High
3	Medium
4	Low

Note: Priority level of each task is incremented (to highest level) periodically over time to avoid task starvation.

Figure 6.10: Adaptive Server Queue Example within Context-Based Negotiation Strategy

The context-based negotiation strategy relies upon server-side logic to analyse the priority level of each incoming computational request and to bid to service the task accordingly. A time-stamping operation is employed by the protocol and is applied to all tasks. This timing mechanism helps ensure lower level tasks avoid starvation by incrementing the priority level of each task as required. The following formula is applied to avoid task starvation:

$$\frac{\textit{Task Service Time Reported to Handheld}}{\textit{Current Expected Task Service Time}} > \begin{array}{c}\textit{Increment Threshold} \\ (\geq 1)\end{array}$$

The *increment threshold* is determined by an administrator to dictate their preferred balance between the speed of response to high urgency computational requests and the service delay they are willing to accept for lower priority tasks. The *current expected task service time* will increase as a task is displaced downwards within an adaptive server's queue of jobs by more urgent tasks. The priority level of the displaced task will increment in the event of the above formula becoming *true*. This leads to a re-adjustment of the *current expected task service time* helping to return the formula to *false*.

6.2.2.1 Evaluation

A medical-based experimental prototype was constructed to evaluate the performance of the proposed context-aware negotiation protocol. The prototype enables a physician to retrieve patient scans that have been image processed in real-time by an adaptive server.

6.2.2.2 Test Case Environment

The test case environment builds upon the architecture presented in Section 6.2.1.2 which details the agent-based negotiation framework. The Place Lab software plug-in is an additional feature which resides within the handheld medical device [Place Lab, 05]. The enhanced environment incorporating Place Lab technology is shown in Figure 6.11. This architectural framework facilitates handheld medical devices and adaptive servers deciding reconfigurable resource allocation through context-based negotiation.

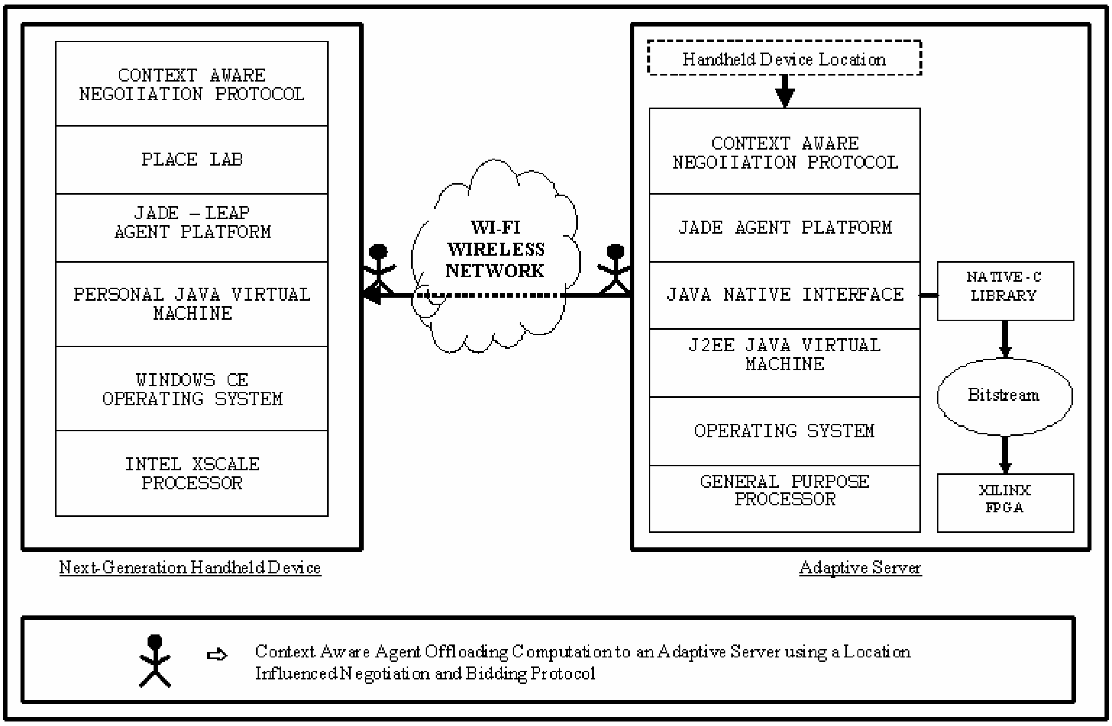

CONTEXT AWARE NEGOIIATION PROTOCOL
PLACE LAB
JADE - LEAP AGENT PLATFORM
PERSONAL JAVA VIRTUAL MACHINE
WINDOWS CE OPERATING SYSTEM
INTEL XSCALE PROCESSOR
Next-Generation Handheld Device
WI-FI WIRELESS NETWORK
Handheld Device Location
CONTEXT AWARE NEGOIIATION PROTOCOL
JADE AGENT PLATFORM
JAVA NATIVE INTERFACE
J2EE JAVA VIRTUAL MACHINE
OPERATING SYSTEM
GENERAL PURPOSE PROCESSOR
NATIVE-C LIBRARY
Bitstream
XILINX FPGA
Adaptive Server
Context Aware Agent Offloading Computation to an Adaptive Server using a Location Influenced Negotiation and Bidding Protocol

Figure 6.11: System Architecture for Context-Based Negotiation Protocol

6.2.2.2.1 Place Lab

A contextual element required for successful deployment of the context-based negotiation protocol is knowledge of the location of the handheld device. This is facilitated within the framework through the incorporation of Place Lab technology. This is an open source development project that uses a radio-beacon based approach to location [LaMarca, 05].

An agent executing on a handheld device can use the Place Lab component to accurately estimate its geographic position. Place Lab has been shown to be able to reliably estimate the location of a portable computer within a range of between fifteen and eighteen meters [Hightower, 04]. The basic GPS[63] scheme provides median accuracy estimates of ten meters [Misra, 99]. GPS usage is limited to outdoor activity whilst its use raises investment and privacy issues [Schilit, 03].

[63] Global Positioning System

In comparison, three primary benefits of incorporating Place Lab technology within device architecture are the increasing proliferation of wireless routers in everyday life, the low capital investment requirements, and the importance placed on maintaining a user's privacy.

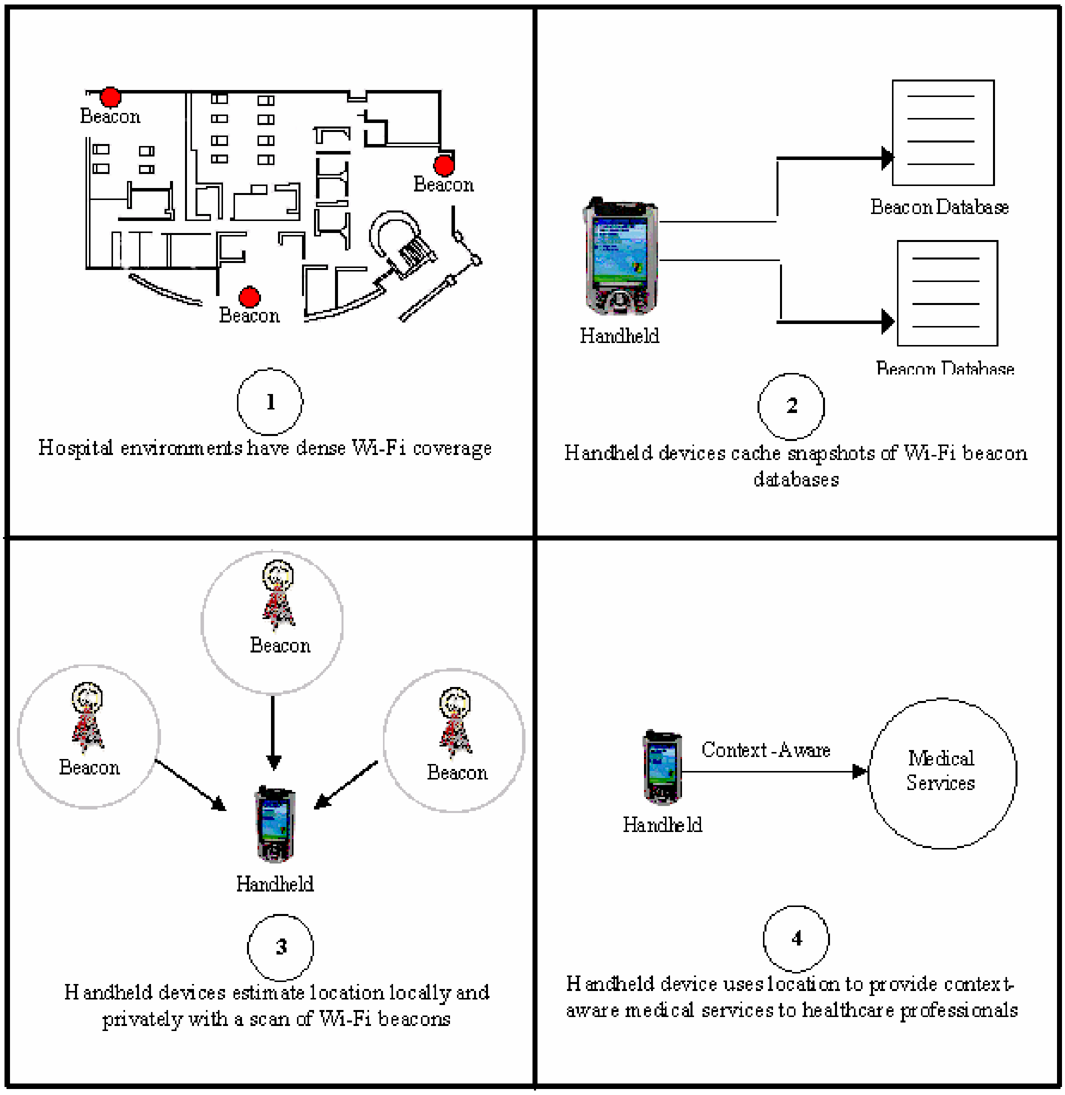

Beacon
Beacon
Beacon
1
Hospital environments have dense Wi-Fi coverage
Beacon Database
Handheld
Beacon Database
2
Handheld devices cache snapshots of Wi-Fi beacon databases
Beacon
Beacon
Beacon
Handheld
3
Handheld devices estimate location locally and privately with a scan of Wi-Fi beacons
Context-Aware
Medical Services
Handheld
4
Handheld device uses location to provide context-aware medical services to healthcare professionals

Figure 6.12: Place Lab Usage Model

The diagram presented in Figure 6.12 highlights the operations executed by Place Lab in delivering geographic location estimates. This outline illustrates a handheld medical device listening for unique identifiers associated with Wi-Fi routers. Each wireless router can be uniquely identified through a MAC[64] address.

[64] Media Access Control

These identifiers are cross-referenced against a cached database of beacon positions to achieve a location estimate.

6.2.2.3 Performance Results

Test cases applied to this experimental prototype evaluate the effectiveness of the proposed protocol in recognising the urgency of a computational request. The context-aware negotiation strategy enhances the quality of service experienced by a handheld user by recognising the importance of their current context.

A test-case simulation examined the effect on service time for a handheld device offloading a computational task using the context-based negotiation protocol. The outcome of this location enhanced approach was contrasted against the agent-based negotiation protocol presented in Section 6.2.1.

6.2.2.3.1 Simulation of Task Priority Levels

The protocol associates the geographic location of a mobile device with a priority level for offloaded computational tasks. The priority levels employed within the test case scenarios relate to a real-world hospital environment and are those presented within Table 6.3.

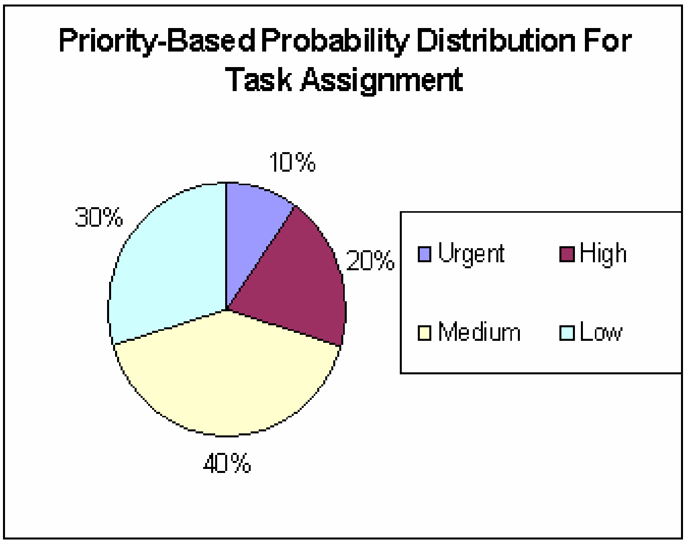


Figure 6.13: Priority-Based Distribution Scenario for Task Assignment

Each task is assigned a priority level using the Monte Carlo random allocation technique. This task assignment uses the priority-based probability distribution shown in Figure 6.13. The generation of a job queue for each adaptive server is simulated using the agent-based negotiation protocol. This protocol ensures the handheld device always offloads its computation to the adaptive server with the smallest queue of jobs. This distribution process generates a state analogous to a load-balanced network as highlighted within Figure 6.5.

6.2.2.3.2 Time to Process Priority-Based Computational Request

The test case results for the time to process a priority-based computational request from a handheld device using the context-based negotiation protocol are presented in Figure 6.14. This service time is dependent upon the urgency of the computational request and the current number of tasks queued for processing by the adaptive servers.

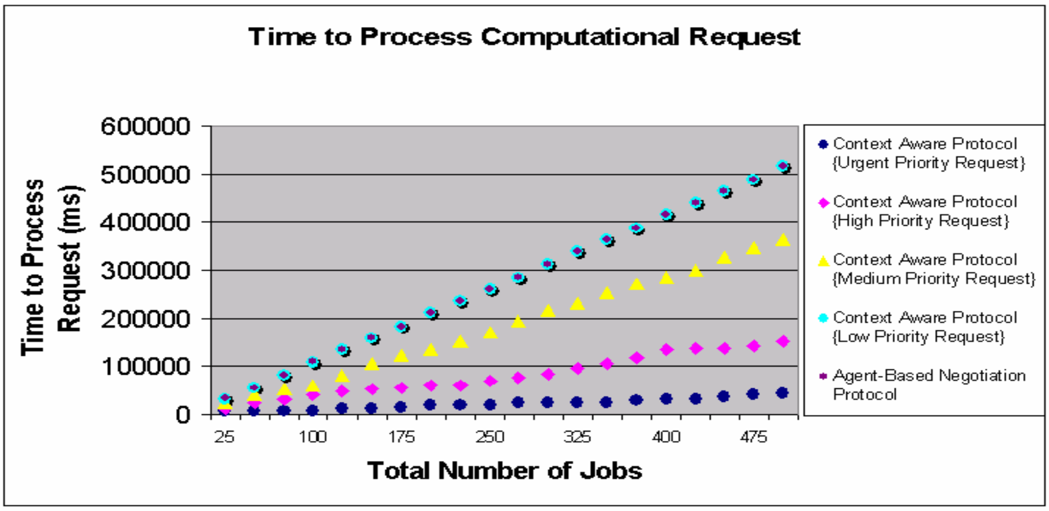


Figure 6.14: Time to Process Computational Requests Using Context-Aware Negotiation Protocol

The request priority is dictated by the location of the handheld device. The service time delivered by the context-based protocol is contrasted against the service time of the agent-based negotiation strategy.

A breakdown of the quantitive time values used for simulation testing within the context-based approach is shown in Table 6.4. These input values were determined through an averaging of results achieved from twenty separate test-runs of both experimental prototype implementations.

Implementation Type	Intercommunication Time	Adaptive Server Execution Time
Context-based Negotiation Protocol	3320 ms	4081 ms
Agent-based Negotiation Model	3110 ms	4075 ms

Table 6.4: Average Quantitive Times for Context-Based Negotiation

The results presented in Figure 6.14 highlight varying degrees of quality of service by the context-based negotiation protocol in servicing different priority-based computational requests. The primary reason for variation in computational request service time is the priority level associated with the task. Task priority dictates the placement of the computational request amongst the adaptive server's queue of jobs. This effect of this job positioning process can be seen clearly in Figure 6.15 which simulates the adaptive server response in placing a task based on the task priority level and the total jobs awaiting process within the adaptive server queue.

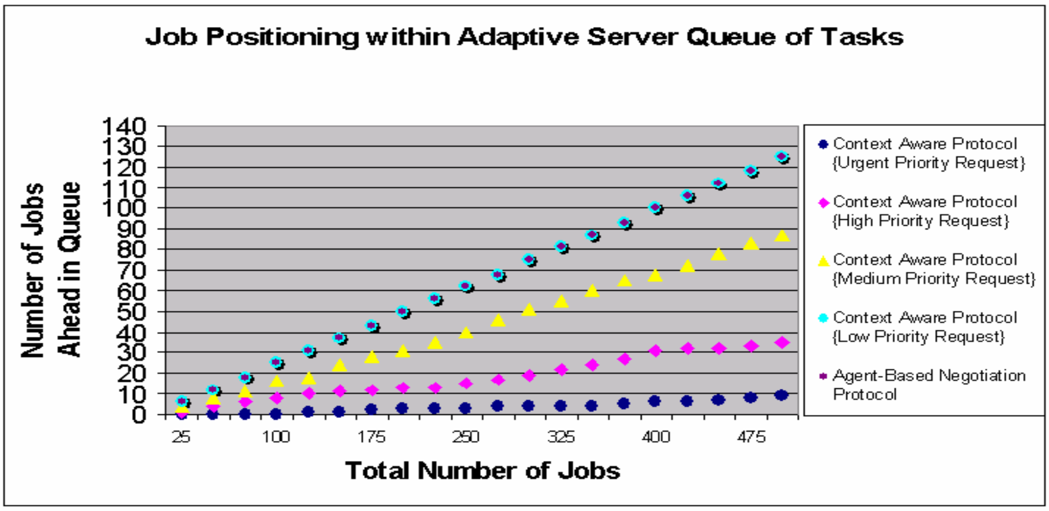


Figure 6.15: Job Positioning within Adaptive Server Queue of Tasks Using Context-Based Negotiation Protocol

The following analysis can be drawn from these service time and job positioning graphs (Figure 6.14 and Figure 6.15 respectively) in terms of adaptive server response to each priority-based computational request:

- Time to Process Low Priority Request

  A low priority computational request from a handheld device is placed at the end of an adaptive server's queue of tasks. This graph shows the time to process a low priority task using the context-based protocol is slightly higher than the time necessary using the agent-based negotiation strategy. This is primarily due to the slight increase in time required for computation within the context-based approach. The additional execution time can be attributed to the complication of computational requests having associated priorities.

- Time to Process Medium, High and Urgent Priority Requests

  The context-based negotiation protocol delivers better performance than the agent-based negotiation strategy for medium, high and urgent priority-based requests. This is primarily due to the ability of the agents operating on adaptive servers to identify the urgency of computational requests and to respond accordingly. The agent-based negotiation strategy recognises all computational requests equally and so is unable to implement a priority-based task queuing system. An example of the context-based approach to recognise request urgency can be seen within the simulation when the *total number of jobs* distributed amongst the adaptive servers is 250. The time to process a new medium computational request from a handheld medical device with this number of tasks awaiting process[65] is 65.6% of the time required with the agent-based approach. The improvement in service time is more sharply observed as the priority level of the task increases. A high-priority and urgent-priority task would

[65] Priority-Based Distribution Scenario for Task Assignment (Figure 6.14) is a determining factor in the evaluation of results.

receive 73.6% and 92.4% quicker service than a task offloaded without priority using the agent-based negotiation strategy.

6.2.2.3.3 Performance Overhead

The primary benefit of associating a priority level with a computational task is that it provides agents operating on adaptive servers with the ability to identify the urgency of each computational request. As the results within Figure 6.15 illustrate, this can clearly improve the quality of service experienced by a handheld device user.

Performance tests presented within Table 6.4 show inter-communication time with the context-based approach takes 6.7% longer on average than the agent-based approach. The adaptive server execution time for a computational task takes 0.15% longer with the context-based approach than the time required for an equivalent computational task with the agent-based approach. These disparities in performance between the context and agent-based negotiation protocols are not vast. The differences do however illustrate the potential for more complicated negotiation protocols to incur additional overhead.

Performance overheads are clearly incurred when providing handheld medical devices with the ability to sense, interpret, and reason about their geographic location. The effects on handheld device resources by the Place Lab software component are shown in Table 6.5 [Pering, 05], [Place Lab, 05].

Type of Performance Overhead	Cost of Performance Overhead
Power Consumption of Wi-Fi Beacon	1.2 Watts (Transmit Mode) 1.0 Watts (Receive Mode)
Memory Requirements	12.28 Mb

Table 6.5: Performance Overhead of Place Lab

6.2.2.3.4 Future Research through Further Refinement

The context-based negotiation protocol could be further expanded by interpreting additional contextual data elements within the handheld device environment. The introduction of additional relevant data could improve decision-making ability within the agent-based framework. This could enhance system capability in recognising the urgency and context of medical practitioner's computational requests.

An example of this enhanced context-based strategy within a telemedicine scenario is presented in Figure 6.16. The proposed computing environment consists of additional technological investment within the field of sensor networks. An agent operating on an adaptive server would determine the priority of a medical practitioner's computational request according to the location of the handheld device and the current state of the patient associated with the request.

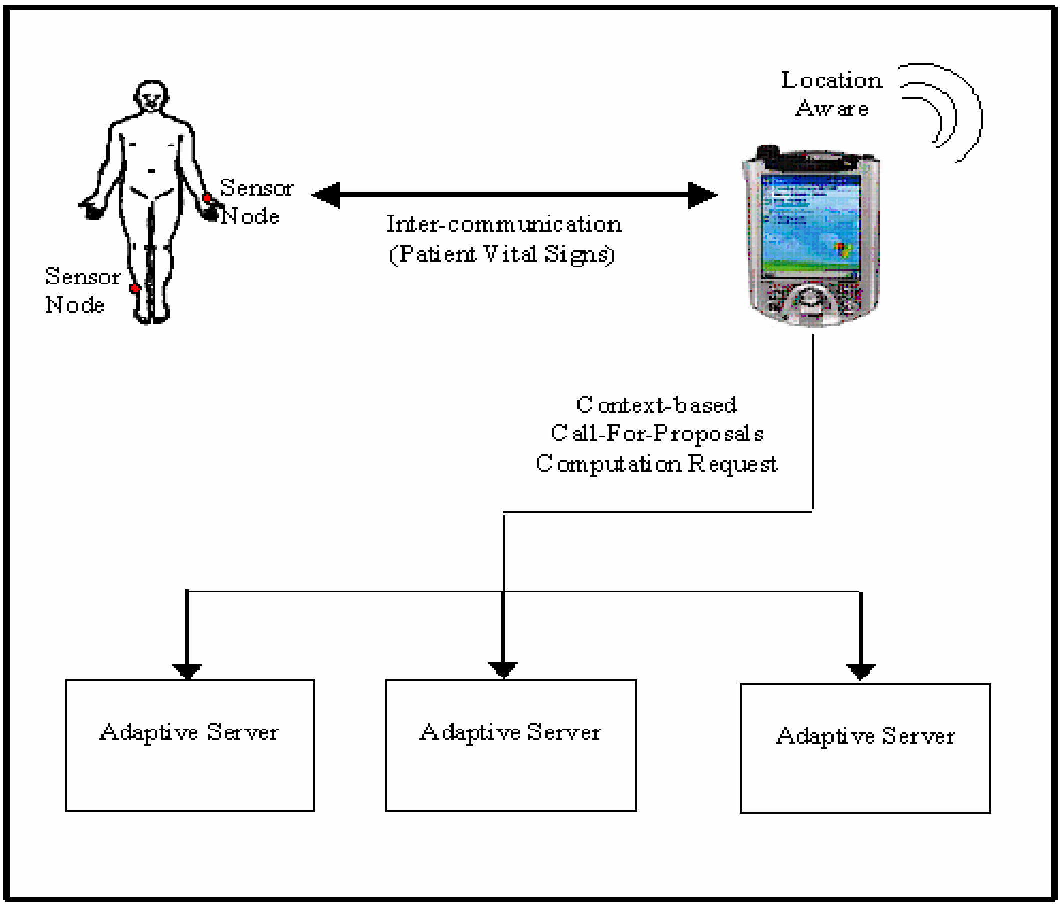


Figure 6.16: Context-Based Negotiation Protocol within Enhanced Telemedicine Scenario

The health status of the patient would be determined in real-time by sensors physically located on the patient's body. This scenario expands upon the initial context-based negotiation protocol proposed within this work by essentially informing decision-making agents with additional data regarding the circumstances of a medical practitioner's computational request.

The potential benefit of this enhanced context-based negotiation protocol can easily be seen in emergency medical situations within a medical environment. A health practitioner may require immediate analysis on medical data relating to a patient who is experiencing a life-threatening situation. The convergence of details relating to the location of the practitioner and the vital signs of the patient helps to signal the medical practitioner's request as highly urgent to all adaptive servers within the environment. This type of telemedicine scenario is possible as wearable sensors become more powerful and ubiquitous [Barton, 02].

6.3 Reconfigurable Hardware as an Integral Handheld Device Component

Reconfigurable hardware can be incorporated as an integral handheld device component. A resource constrained portable computer can adapt its reconfigurable hardware dynamically. The programmable logic effectively provides application diversity by allowing a handheld device to modify its reconfigurable hardware according to the specific requirements of each executing application. This ability of a portable computer to dynamically adjust programmable logic resources when switching between applications is analogous to the concept of cache management.

General advantages of incorporating reconfigurable hardware as an integral handheld device resource can be summarised as follows:

- Firstly, it is an enabling technology that allows mobile devices to execute a broad range of performance-intensive applications. Essentially, the reconfigurable substrate delivers computational power and diversity whilst respecting the physical constraints of the handheld system.
- Secondly, it allows a vendor to release early product versions on time ensuring they comply with their time-to-market constraints. Full product functionality can then be achieved in the market-place by distributing reconfigurable hardware-based patches at a future date.
- Thirdly, it allows manufacturers to disseminate additional reconfigurable hardware-based applications after consumer purchase. These can be configured and executed on the handheld system. This facet of reconfigurable technology can increase both product lifetimes for the consumer and revenues for the vendor.

Clearly, reconfigurable hardware has the potential to dynamically increase the system functionality and versatility of a handheld device without major loss in performance. The general advantages outlined above merely hint at the range of benefits that can be derived through integrating the technology as an integral component within a portable computer. However, mobile systems need a sophisticated middleware framework to help them effectively utilise these integral reconfigurable resources.

An agent-based middleware is the environment employed for handheld device management. Agent technology allows for coherent representation and communication of device characteristics, user preferences and application requirements. Agents are efficient in their use of network bandwidth and can deal with intermittent wireless connections.

Additionally, they are capable of coherently handling the execution and transmission of reconfigurable hardware-software based computations. Significantly, from a development perspective, agent computing lends itself favourably towards the design and implementation of heterogeneous mobile device environments. It encourages a high-level of abstraction and this is beneficial in conceptualising and constructing these complex distributed systems [Jennings, 01].

The following two deployment strategies outline the ability of agent middleware to empower handheld devices to effectively exploit their reconfigurable hardware components:

- Push-Based Configuration Management Strategy
- Context-Aware Reconfiguration of Handheld Devices

These deployment strategies extend and enhance our initial development of an application solutions retrieval protocol [O' Sullivan, 04c]. This protocol enables a handheld device to dynamically retrieve reconfigurable hardware/software-based application solutions from a provisioning server using agent technology.

6.3.2 Push-Based Configuration Management Strategy

Distribution of product updates to handheld devices increases product lifetimes for consumers and also has the potential to increase revenues and brand loyalty for vendors. Dynamic provisioning of these application solutions to handheld devices is complex due to their heterogeneous nature. Product updates should be composed of both software and reconfigurable hardware code which are tailored to the physical constraints of the device.

Increasingly intense competition combined with time-to-market pressures has also compelled mobile device manufacturers to release early product versions to guarantee adequate market share. Full-product functionality is achieved in the market-place through a configuration management technique of distributing reconfigurable hardware-software based updates and patches. Push technology is a distribution mechanism initiated by the vendor enabling delivery of product updates to handheld devices.

Portable systems are evolving and clearly demand more proficient development methods and tools for their design, deployment and management [Fleischmann, 99]. A flexible, robust and proactive distribution framework is especially required to allow vendors confidently disseminate their updates and modifications to customers after initial product releases. All updates should interoperate seamlessly with the handheld device ensuring minimal disruption to the customer.

This deployment strategy proposes a configuration management architectural framework incorporating a mobile agent based push methodology for networked reconfigurable handheld devices [O' Sullivan, 04a]. Agent technology is particularly suitable as a push-based distribution mechanism for mobile systems. A push strategy utilising agent concepts allows for the coherent distribution of hardware-software application solutions to networked handheld devices. An agent-based middleware framework is efficient in its use of network bandwidth and is flexible in dealing with intermittent network connections. These characteristics are highly beneficial for mobile wireless devices. Agent technology encourages management decentralisation which minimises the load on the network management centre and reduces points of failure within the network [Raibulet, 00].

6.3.2.1 Review of Push Technology for Data Dissemination

Push technology has been applied within various research communities as an architectural paradigm for data dissemination.

Telecommunications research has explored the push concept as a method for the provisioning of configuration settings to mobile phones [Ladas, 01]. The WAP[66] and the GSM[67] short messaging system[68] are combined to provide the implementation framework. A beneficial scenario is highlighted whereby the hand-held devices belonging to selected personnel at a disaster site are automatically adapted for specifically directed emergency communications.

Parallel computing has also been exposed to the potential of push-based technology. A novel mobile agent-based push methodology has been proposed within the supercomputing domain [Xu, 00]. This approach allows users to dispatch their jobs as agents who roam the network seeking servers on which they can execute their task. This is advantageous as it produces an adaptive and fault tolerant execution model.

Push-based technology also plays a key role within the configuration management community [Hall, 98]. A push-based distribution strategy enabling software deployment utilising a mobile agent framework is presented. Key benefits of an agent-based approach to configuration management are flexibility, reliability and increased performance [Berghoff, 96].

An embedded systems configuration management technique focusing on hardware upgrades has also been proposed [Casselman, 02]. Their methodology targets remote FPGA devices. Object-oriented programming techniques are applied to an

[66] Wireless Application Protocol

[67] Global System for Mobile Communication

[68] Also know as SMS

FPGA configuration bitstream. An object is created encapsulating any additional information it may require for its delivery, verification and use. The object is packaged into a payload which is pushed over a TCP/IP network to the embedded device. Through object-oriented design and analysis techniques, an additional layer of abstraction is achieved providing a more robust and reliable deployment methodology.

It is clear the concept of configuration management is increasing in importance within the embedded device co-design community. It is recognised that with shorter time-to-market windows and increasing demands for additional product functionality, there is a need to develop methods and tools to dynamically deploy hardware-software based application solutions to consumer embedded devices [Fleischmann, 99].

The approaches of Hall and Berghoff show the combination of mobile agents and push-based distribution strategies are a highly appropriate implementation methodology for configuration management. The push-based configuration management strategy proposed extends these approaches to target the unique properties of reconfigurable handheld devices to enable cohesive upgrades to their reconfigurable hardware and software based components.

Agent Name	**Agent Role**
Mobile Device Manager	*<u>All Scenarios</u>* This single instance agent is a permanent resident on the mobile medical device and has responsibility for gathering and maintaining information about the physical device and its owner. The agent operates as the main point of contact between the user and the provisioning server architecture. *<u>Push-Based Configuration Management Strategy</u>* Application Courier agents interact locally with the Mobile Device Manager to ensure successful execution of an update process. *<u>Context-Aware Reconfiguration of Handheld Devices</u>* The Mobile Device Manager registers with the Provisioning Server Manager to receive application solutions depending upon the context of the user of the handheld device.
Directory Facilitator (DF)	*<u>All Scenarios</u>* This agent is responsible for maintaining knowledge about the location and services of each agent within the platform. *<u>Push-Based Configuration Management Strategy</u>* This information is used by the Distribution Master to determine the destination of each Application Courier. *<u>Context-Aware Reconfiguration of Handheld Devices</u>* The Mobile Device Manager on the handheld device uses the directory facilitator service to discover the location of the provisioning server manager. This information is used to register to receive propagated application solutions. The Distribution Master on the provisioning server also uses this service to determine the destination of the Application Courier agent delivering an application to a portable computer.

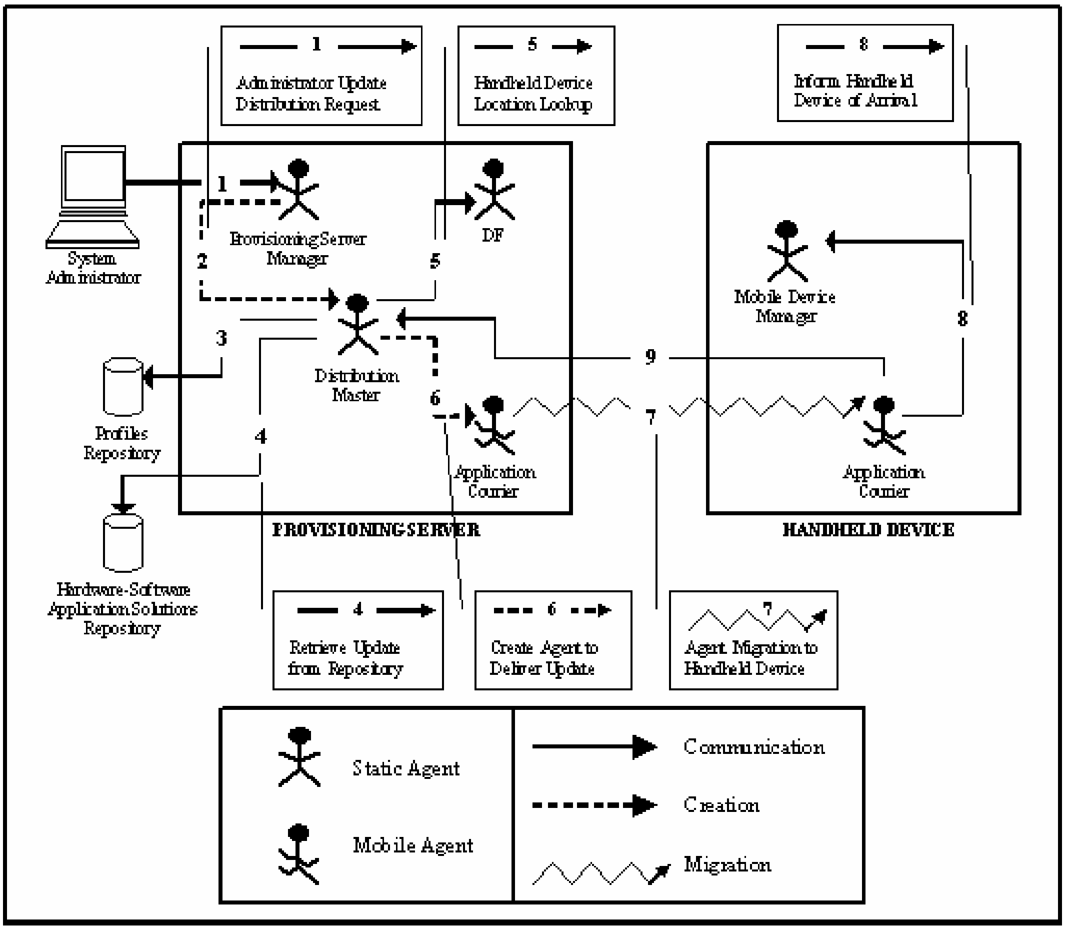
Distribution Master	<u>*All Scenarios*</u> This agent is instantiated as needed and has responsibility for coordinating deployment of application solutions to all appropriate handheld medical devices. <u>*Push-Based Configuration Management Strategy*</u> It coordinates with a profiles repository to discover all devices that qualify for a solution. The update is retrieved from a hardware-software application solutions repository. This agent creates a number of Application Courier slaves to deliver the update. The functionality of this agent is based on the master component of the master-slave design pattern which is outlined in further detail in Section 6.3.2.3. <u>*Context-Aware Reconfiguration of Handheld Devices*</u> The Distribution Master determines the location of the handheld device necessitating an application solution. It creates an Application Courier to carry the application to the portable computer.
Provisioning Server Manager	<u>*All Scenarios*</u> Instances of this agent reside on each provisioning server. It operates as the main point of contact between the system administrator and the network infrastructure. <u>*Push-Based Configuration Management Strategy*</u> It responds to distribution requests by the administrator by invoking a Distribution Master agent to communicate product updates or fixes to all relevant embedded devices. <u>*Context-Aware Reconfiguration of Handheld Devices*</u> This agent delegates decision-making regarding the propagation of application solutions to the Expert System Manager. It passes contextual data associated with each portable computer to the Expert System Manager periodically. The Provisioning Server Manager creates a Distribution Master to handle the propagation of an application solution to a handheld device.
Expert System Manager	<u>*Context-Aware Reconfiguration of Handheld Devices*</u> The Expert System Manager maintains an interface to a rule-based expert system. This agent is responsible for controlling and interacting with the

	rule engine. This involves gathering the contextual data elements of a handheld device and communicating these values to the expert system. The decision of the rule engine informs the Expert System Manager whether application management operations are required.
Application Courier	*All Scenarios* This agent is instantiated as needed and has responsibility for carrying application solutions to handheld medical devices. The agent encapsulates all necessary software and reconfigurable hardware code to execute an update on the mobile device. *Push-Based Configuration Management Strategy* It provides a coherent approach to push-based distribution. The functionality of this agent is based on the slave component of the master-slave design pattern. *Context-Aware Reconfiguration of Handheld Devices* The Application Courier encapsulates an application solution to be propagated to a portable computer. The agent interacts with the Mobile Device Manager upon arrival and informs the user of the new application.

Table 6.6: Agent Roles and Responsibilities within Integral Handheld Device Resource Deployment Scenarios

6.3.2.2 Architectural Overview

The agent architecture developed to facilitate this push-based configuration management strategy is shown in Figure 6.17. This diagram highlights paths of intercommunication amongst agents as well as dynamic agent creation. The architecture was developed using the Gaia methodology [Wooldridge, 99]. An overview of this agent-oriented analysis and design methodology is presented in Section 4.6.1. The role of each agent is outlined within Table 6.6. This table can be used as a reference guide to Figure 6.17.



1
Administrator Update Distribution Request
5
Handheld Device Location Lookup
8
Inform Handheld Device of Arrival
System Administrator
1
2
Provisioning Server Manager
DF
5
3
Distribution Master
6
Application Courier
Profiles Repository
4
Hardware-Software Application Solutions Repository
9
7
Mobile Device Manager
8
Application Courier
PROVISIONING SERVER
HANDHELD DEVICE
4
Retrieve Update from Repository
6
Create Agent to Deliver Update
7
Agent Migration to Handheld Device
Static Agent
Mobile Agent
Communication
Creation
Migration

Figure 6.17: Agent Architecture Implementing Push-Based Configuration Management Strategy

The above mobile agent-based push framework allows for the coherent distribution of application solutions to handheld medical devices. An agent can completely encapsulate all necessary reconfigurable hardware and software code comprising an application solution.

6.3.2.3 Master-Slave Design Pattern

Design patterns provide a strong foundation in constructing agent-oriented applications. The *push-based configuration management strategy* employs the master-slave design [Buschmann, 95]. This behavioural pattern is presented in Figure 6.18. The pattern is appropriate for frameworks replicating a particular service through delegating the same task to several independent suppliers.

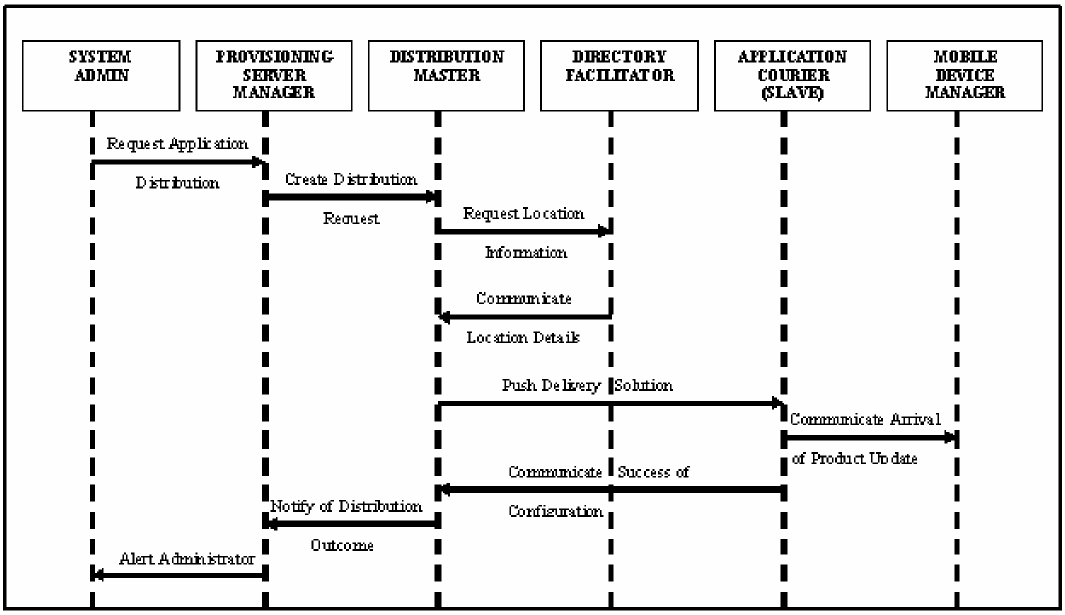

SYSTEM ADMIN
PROVISIONING SERVER MANAGER
DISTRIBUTION MASTER
DIRECTORY FACILITATOR
APPLICATION COURIER (SLAVE)
MOBILE DEVICE MANAGER
Request Application Distribution
Create Distribution Request
Request Location Information
Communicate Location Details
Push Delivery Solution
Communicate Arrival of Product Update
Communicate Success of Configuration
Notify of Distribution Outcome
Alert Administrator

Figure 6.18: Master-Slave Behavioural Pattern

The pattern is mostly applied to industrial systems requiring fault tolerance, however it is also a highly apt framework for the distribution of application solutions to networked handheld medical devices. The Distribution Master and Application Courier agents represent the master and slave components of the pattern respectively.

6.3.2.4 Evaluation

A prototype system was implemented to determine the feasibility of the *push-based configuration management strategy*. The evaluation system propagates image processing applications to a medical-based portable computer using a distribution process triggered by a system administrator.

6.3.2.5 Test Case Environment

The test case environment incorporates reconfigurable hardware as an integral resource of a portable computer. The handheld device employed within this test case was a Dell Inspiron 510m Notebook [Dell, 05b]. A detailed account of the reasoning behind choosing this device is provided within Section 6.4. This handheld device executes the JADE agent platform. The provisioning server operates on a high-end Pentium PC also running the JADE agent platform.

An architectural diagram of the reconfigurable hardware-based mobile medical device employed with prototype development is presented in Figure 6.19.

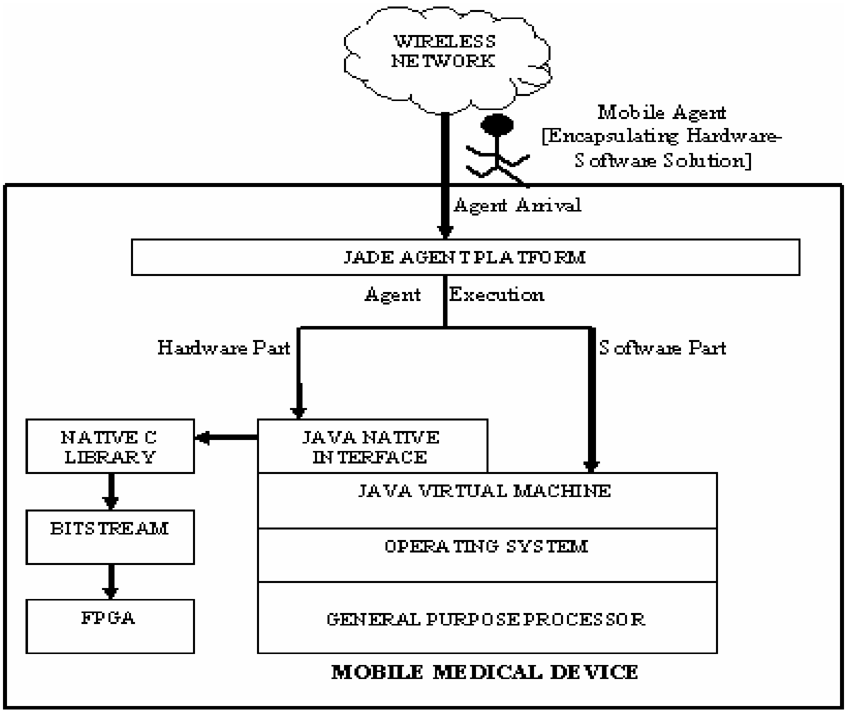


Figure 6.19: Mobile Medical Device System Architecture

The coherent distribution of application solutions is a primary benefit of the *push-based configuration management strategy.* This protocol enables system administrators to propagate cohesive upgrades to reconfigurable handheld devices.

Test-case analysis of intercommunication and execution times for agent-based and client-server based implementation models was obtained and analysed in Section 6.2.1.3.2. Intercommunication time is perceived to be of secondary importance within the *push-based configuration management strategy* as product updates and fixes are irregular and infrequent events. The relatively slow execution time of the agent-based model in comparison with the client-server model is a concern. A design mechanism conceived to improve execution time is presented in Section 4.8.

6.3.2.6 Future Work

Security will play a crucial role in achieving a viable configuration management methodology for handheld mobile devices. Agent characteristics of mobility and autonomy heighten issues of security within mobile agent-based frameworks. Security aspects to be considered are outlined in Table 6.7.

Security Feature	**Description**
Confidentiality	Ensure that only the intended party can read the information.
Integrity	Ensure that information cannot be modified.
Authentication	Ensure the identity of an entity within the framework.
Authorisation	Grant access rights based on the identity of an entity.
Non-repudiation	Ensure an entity cannot deny involvement in a previous commitment or transaction.

Table 6.7: Security within Handheld Device Configuration Management

A security enhancement to the configuration management infrastructure addresses safety concerns using the Jade-S security plug-in for agent frameworks [Vitaglione, 02b].

Jade-S is based on the Java security model [Java, 05c] and incorporates the following technologies:

- Java Authentication and Authorisation Service (JAAS)
  This provides the capability to enforce authentication and access control upon principals [Sun, 05a].

- Java Cryptography Extension (JCE)
  This provides a range of encryption algorithms and also allows for key generation and management [Sun, 05b].

- Java Secure Socket Extension (JSSE)
  This allows for secure communications using channels constructed with Secure Sockets Layer (SSL) and Transport Layer Security (TLS) technology [Sun, 05c].

Each agent within the distribution framework has an associated identity certificate. This provides guarantees as to the identity and ownership of agents. Requests by agents for permission to execute actions are granted or denied by the Java security manager. The decision is made on the basis of their list of allowed privileges in the system policy file. An agent can enhance its capabilities with a delegation certificate. This is a document attesting the necessary authorisation to execute determined actions on behalf of others [Poggi, 01].

The Jade-S framework is an additional feature which can be incorporated into the *push-based configuration management protocol* to ensure confidentiality and integrity of all product update operations.

6.3.3 Context-Aware Reconfiguration of Handheld Devices

This deployment strategy seeks to enhance utilisation of reconfigurable resources contained within a mobile device. The strategy proposes using a context aware agent-based framework to push reconfigurable hardware and software based application solutions to a handheld device [O'Sullivan, 06b].

This protocol is related to the *push-based configuration management strategy* (ref. Section 6.3.2) as it also distributes applications through a server-initiated process. The *context-aware protocol* also borrows from the CAMMD project (ref. Section 5.1.1) by employing the concept of context in the decision making process.

As shown, the *push-based configuration management strategy* focuses upon the dynamic provisioning of reconfigurable hardware based product updates to handheld devices. This *context-aware protocol* increases the frequency of configuration by seeking to push reconfigurable hardware applications to a mobile device depending on the contextual environment of a user. This enables a portable device to have its hardware configuration dynamically tailored to the usage (i.e. application) requirements of a user.

The contextual elements required to enable effective configuration management are the location of the mobile device, the time of day, and the activity of the user. These elements are interpreted by a rule-based expert system operating on a provisioning server to determine if a reconfigurable hardware application should be pushed to a portable computer.

An example scenario of this deployment strategy within a distributed telemedicine environment for a medical practitioner is shown within Table 6.8. The user activity is derived from a predetermined schedule of user appointments and their associated application preferences.

Reconfigurable Hardware Application Pushed to Mobile Device	Mobile Device Location	Time	User Activity	Reason
Image Processing Application	Hospital Ward	9 AM	Patient Check-up	Practitioner analyses patient scans using an image processing application.
Speech Recognition Application	Office	1 PM	Taking Patient Assessment Notes	Practitioner takes notes on patients using a voice recognition system.
Video Processing Application	Home	7 PM	Personal Time	Practitioner likes to watch films on mobile device whilst relaxing at home.

Table 6.8: Example Medical Scenario of Context-Aware Hardware Reconfiguration of Handheld Device

6.3.3.1 Architectural Overview

The agent architecture developed to facilitate this context-aware reconfiguration strategy is shown in Figure 6.20. This diagram highlights paths of intercommunication amongst agents as well as dynamic agent creation. The architecture was developed using the Gaia methodology [Wooldridge, 99]. An overview of this agent-oriented analysis and design methodology is presented in Section 4.6.1. The role of each agent is outlined within Table 6.6. This table can be used as a reference guide to Figure 6.20.

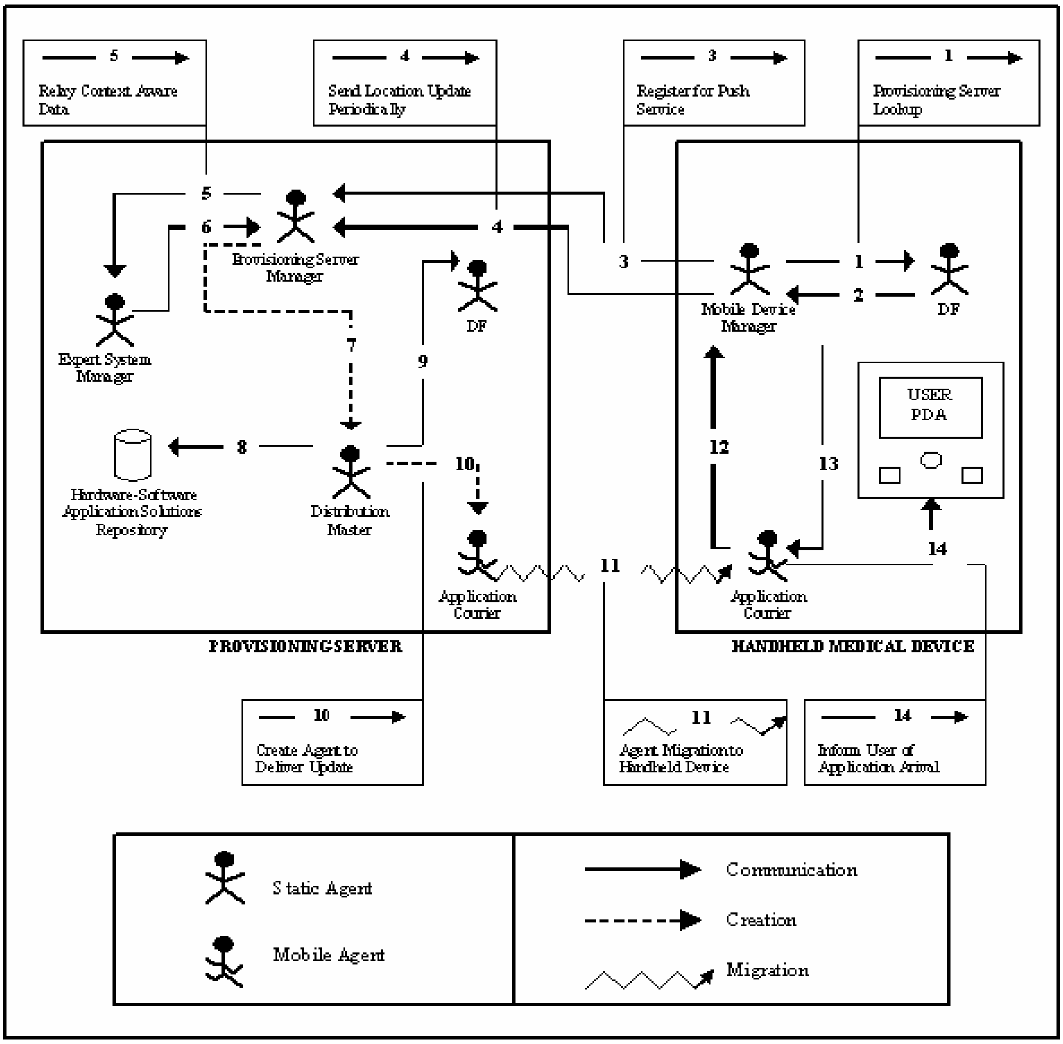


Figure 6.20: Agent Architecture Implementing Context-Aware Reconfiguration Protocol

The agent architecture employs static and mobile agents to realise the context-aware reconfiguration protocol. An *application courier* mobile agent migrates from the provisioning server to the handheld medical device encapsulating an application solution. An overview of mobile agents detailing the benefits of employing them within telecommunication environments is presented in Section 4.3.

6.3.3.1 Evaluation

A medical-based experimental prototype was constructed to evaluate the performance of the context-aware reconfiguration protocol. This prototype follows closely upon the evaluation framework developed for the CAMMD system (ref. Section 5.1.1).

The evaluation system propagates image processing applications to a medical-based portable computer depending on the contextual environment of the physician operating the device.

6.3.3.2 Test Case Environment

The test case environment builds upon the architecture presented in Section 5.1.3 which details the CAMMD deployment architecture. The incorporation of reconfigurable hardware as integral resource of the portable computer is an additional element. This additional feature enables a device to have its hardware configuration dynamically tailored to the usage (i.e. application) requirements of a user.

The handheld device employed within this test case was a Dell Inspiron 510m Notebook [Dell, 05b]. A detailed account of the reasoning behind choosing this device is provided within Section 6.4. This handheld device executes the JADE agent platform. The provisioning server operates on a high-end Pentium PC also running the JADE agent platform.

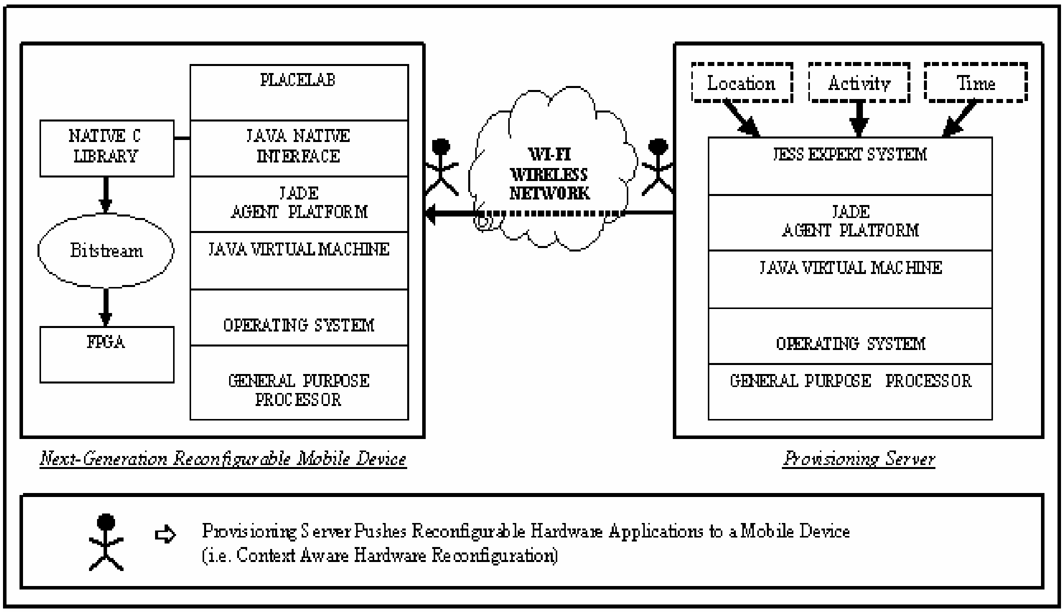


Figure 6.21: Context-Aware Reconfigurable System Architectural Framework

A Jess rule-based expert system resides on the provisioning server. Reconfigurable hardware bitstreams propagated to handheld devices within the hospital scenario are stored in a flat-file repository. A Place Lab plug-in resides on each handheld device enabling an accurate location estimate to be communicated to the provisioning server. Agents communicate between the distributed components over a Wi-Fi network.

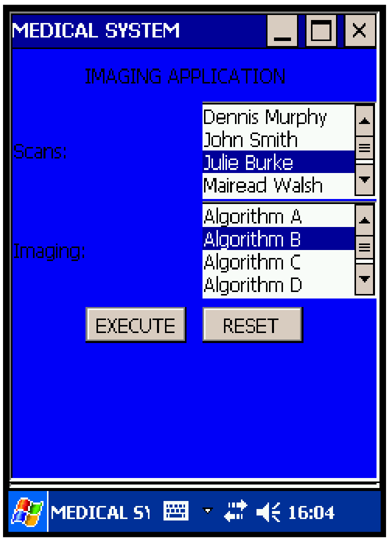
The components required to implement the *context-aware reconfiguration protocol* are presented within the system architectural framework in Figure 6.21. This diagram also highlights the conceptual context-aware elements of location, activity, and time which are necessary for successful protocol execution.

6.3.3.3 Performance Results

The testing scenario employed for the *context-aware reconfiguration protocol* closely follows the methodology used for the CAMMD framework (ref. Section 5.1.4). A *usability and interaction* test dealing with user navigation was examined with regard to the *context-aware reconfiguration protocol*. This test compares the proposed agent protocol against an RMI medical-based implementation.

The simulation environment of ten geographically distributed wards conceived within CAMMD testing was utilised. A hypothetical use-case scenario was conceived which required practitioners to execute image processing applications on their mobile medical devices during patient consultations.

These image filtering applications enable a physician to perform image analysis on patient scans in real-time. The hypothetical scenario associated an image processing algorithm from a list of ten algorithms with each individual patient. The patient to ward distribution can be referenced in Table 5.3.



MEDICAL SYSTEM

IMAGING APPLICATION

Scans: Dennis Murphy, John Smith, Julie Burke, Mairead Walsh

Imaging: Algorithm A, Algorithm B, Algorithm C, Algorithm D

Figure 6.22: Context-Aware Reconfiguration Protocol Screenshot

A walk-through of the first five wards was conducted by ten individuals to achieve results for the test case. The *usability and interaction* test evaluated the average time required by each user to select both the appropriate image processing algorithm and patient scan for each consultation. This test case examined the usability of both implementations. A screenshot of this selection interface is shown in Figure 6.22 and the results of the test case are shown in Figure 6.23.

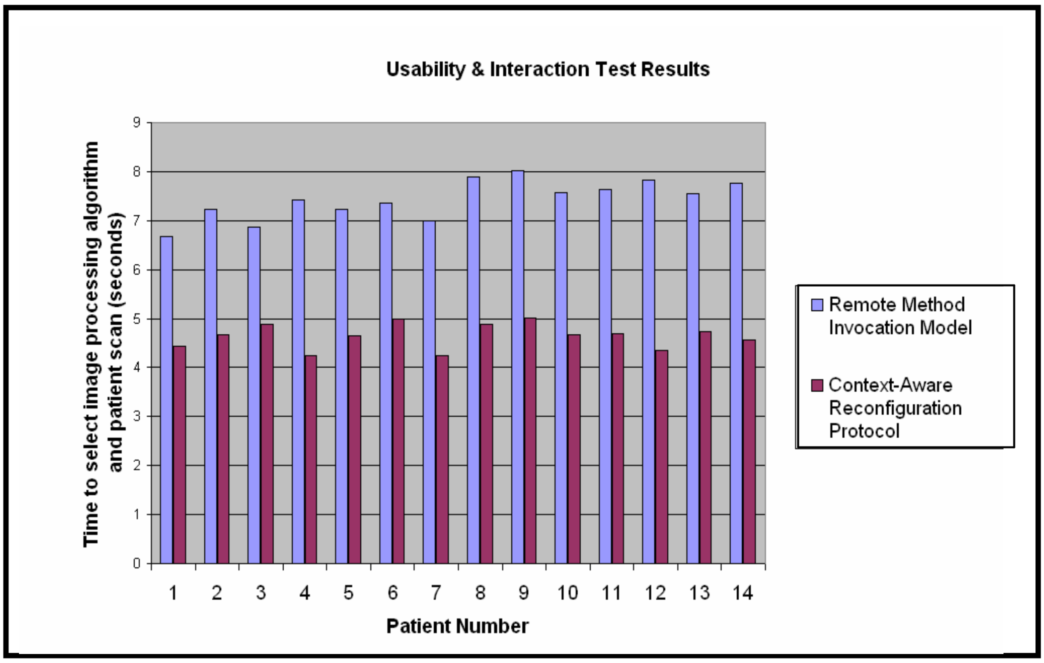


Figure 6.23: Usability and Interaction Test Results

The handheld device executing the *context-aware reconfiguration* protocol benefited from the concise nature of the lists returned from the provisioning server. These lists contained image processing algorithms and patient scans. This conciseness allowed faster navigation for instructing the device to perform image processing computations. The navigation time with the RMI-based implementation was on average approximately 2.79 seconds slower. A primary cause of this delay is due to the extra time required to locate and associate an image algorithm with a patient scan when dealing with larger lists. The *context-aware reconfiguration* protocol clearly improved user interaction by helping to avoid this information overload.

Another primary benefit of the agent-based protocol is its ability to optimise handheld device storage. Bitstreams representing reconfigurable hardware implementations of imaging algorithms are propagated to portable computers along with their associated patient scans. This is possible due to the conceptual relationship established between patient scans and image processing algorithms. A bitstream implementing an edge detection algorithm for a 256*256 pixel image is approximately 323Kb in size. It is acceptable for the handheld medical device to store two or three algorithms at any one time, however it is not feasible for the device to maintain all possible bitstream configurations of image processing algorithms in memory. The *context-aware reconfiguration* protocol clearly helps optimise memory constraints of a portable computer and this improves device portability.

6.4 Handheld Device Usage

A Dell Inspiron 510m mobile notebook [Dell, 05b] was employed to develop the agent development systems which incorporated reconfigurable hardware as an integral device resource. The Dell Axim mobile device [Dell, 05a] was initially examined to determine its suitability to act as a networked reconfigurable portable computer.

However, a number of technological interface issues were encountered with this device. The Dell PDA has a 16-bit Windows CE operating system which was incompatible with the 32-bit windows device libraries (ref. Section 3.6.2.2) developed by Celoxica Corporation for interaction with their reconfigurable hardware boards. Celoxica Corporation was also unwilling to release the source code to their device libraries.

The JBits XHWIF API (ref. Section 3.6.2.1) is operational with the Windows CE operating system, however the Xilinx Virtex board compatible with the JBits device drivers is PCI-based and so physical connection with the Dell PDA is difficult. Tablet PCs were also investigated as a potential solution, however financial constraints nullified this option. Employing a Tablet PC was the preferred approach and the potential of these devices to act as mobile reconfigurable hardware platforms is highlighted within the research of the Enamorado project group [Nikolouzou, 04] and [Kosmatos, 04].

The technological shortcoming of using a mobile notebook instead of a more computationally deficient handheld device is considered minor as the development prototype systems are primarily focused upon the effectiveness of the proposed middleware protocols. The overall analysis of their operation and value is applicable regardless of the technical specification of the reconfigurable handheld device.

CHAPTER 7

Exploiting Learning and Collaboration Characteristics of Agents

7.1 Introduction

The following deployment strategies outline the ability of agent middleware to empower handheld devices to more effectively exploit surrounding adaptive server technology:

- Learning and Adaptation Negotiation Protocol
- Collective Past-Experience Learning Strategy

7.2 Learning & Adaptation Negotiation Protocol

Learning and adaptation are rudimentary characteristics that can be embodied into an agent[69]. These attributes enable an agent to learn about its environment and to adapt to conditions within it. This ability of an agent to gather knowledge about neighbouring entities and to use this information to guide future decision-making is a powerful trait. It that can be extremely beneficial for the individual on whose behalf the agent is acting. The integration of learning and adaptation characteristics into an agent-based framework can enable mobile medical devices to more effectively utilise surrounding reconfigurable resources [O' Sullivan, 06a].

Learning and adaptation capabilities have been introduced into the participating agents within the agent-based negotiation protocol outlined in Section 5.2.1. These agent characteristics can enhance the decision-making process on a mobile medical device regarding adaptive server utilisation. Agent decisions concerning computational task offloading become more informed through the utilisation of knowledge gathered from past interactions with surrounding adaptive servers.

The agent architecture developed to facilitate this learning and adaptation technique employs the same model as the simplified agent negotiation protocol

[69] Identifying characteristics of agents are described in detail in Section 4.2

shown in Figure 6.1. This diagram highlights paths of intercommunication amongst agents as well as dynamic agent creation. The architecture was developed using the Gaia methodology [Wooldridge, 99]. An overview of this agent-oriented analysis and design methodology is presented in Section 4.6.1.

The additional complexity required for successful operation of the learning and adaptation protocol is contained within the responsibilities and actions of the participating agents. Details regarding the activities of each agent within the learning and negotiation protocol are outlined within Table 7.1. This table can be used as a reference guide to Figure 6.1.

Agent Name	Agent Role
Mobile Device Manager	This single instance agent is a permanent resident on the mobile medical device and has responsibility for gathering and maintaining information about the physical device and its owner. The agent operates as the main point of contact between the user and the distributed adaptive server architecture. It predicts or responds to resource limitations on the mobile medical device by attempting to schedule a performance intensive computation upon a neighbouring adaptive server. The mobile device manager agent initialises the process of selection through invoking a resource requester agent.
Directory Facilitator	This agent is responsible for maintaining knowledge about the location and services of each agent within the platform.
Resource Requester (Negotiator)	The Resource Requester is the key agent entity within the learning and adaptation negotiation protocol. It implements the *Performance Analysis, Knowledge Building, Knowledge Acquisition, and Informed Action* stages of the protocol. This agent is instantiated as needed and is responsible for initiating the negotiation process with adaptive servers for access to their reconfigurable resources. This negotiation strategy employs concepts based upon the contract-net protocol [Smith, 80]. The resource requester agent initially retrieves a list of all adaptive manager agents within the network from the directory facilitator. A *call-for-proposals* computation request is broadcast to all adaptive manager agents on this list. The resource requester then evaluates all adaptive manager bids to determine the best offload option. This entails retrieving the *Bid Adjustment* ratings of each adaptive server and applying it to their respective bids. The Resource Requester accesses a flat file repository to retrieve and update the *Bid Adjustment* ratings of each adaptive server. The computational task is assigned to the adaptive server which promises to service the request in the quickest time (taking into account the accuracy of their bidding history). The Resource Requester updates the *Bid Adjustment* rating associated with the adaptive server upon receiving the result of a computational task.

Adaptive Manager (Negotiator)	This agent is responsible for facilitating access to reconfigurable resources on an adaptive server. The adaptive manager attempts to successfully bid to service a mobile medical device's computational request. Upon receiving a *call-for-proposals*, an adaptive manager examines its current queue of jobs and estimates their total service time. The result of this evaluation combined with an estimate of the time required to service the current computation request determines the adaptive manager bid.
Repository Handler	This agent has responsibility for retrieving the reconfigurable bitstream representation of an algorithm required for a mobile device's computational request. The agent is also responsible for returning any additional data that may be required e.g. scanned patient images.
Reconfigurable Resource Handler	This agent is responsible for the process of downloading the bitstream configuration to the reconfigurable resource, interacting with the FPGA and communicating the results of the hardware computation to the adaptive manager.

Table 7.1: Agent Roles and Responsibilities within Learning and Adaptation Negotiation Protocol

Learning and adaptation within the protocol is confined to client-side operations. The four stages within this pseudo-intelligent technique are shown in Figure 7.1 and are detailed below:

- Performance Analysis

  An analysis mechanism is employed by the participating agent on the mobile medical device to enable it to evaluate the performance of an adaptive server to which it has offloaded a computational task. A timing technique is utilised by the agent to allow it to effectively critique service provided by an adaptive server. This analysis is achieved in terms of comparing an adaptive server's actual performance against the performance originally promised in their service bid. Potential reasons for poor bid estimation by an adaptive server are presented in Table 7.2.

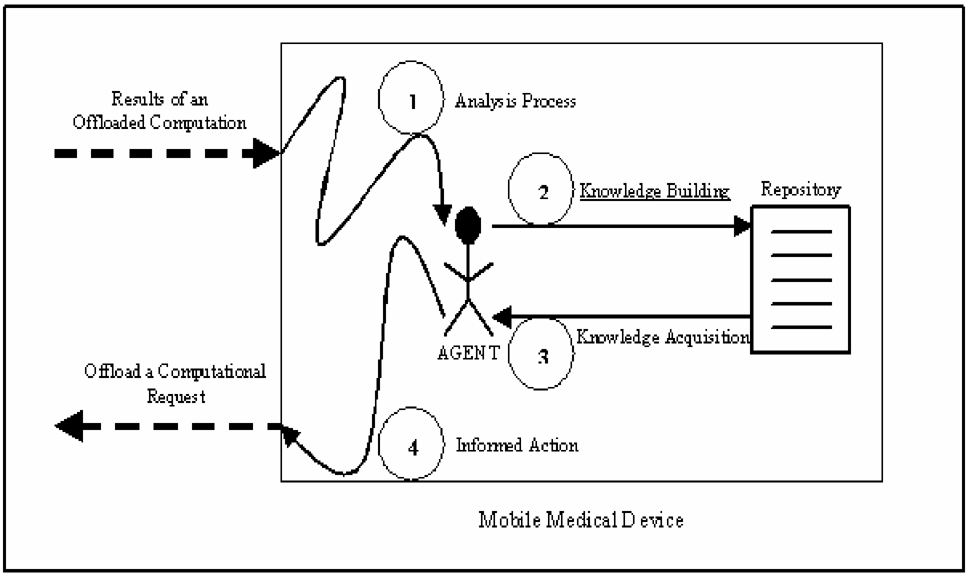


Figure 7.1 Pseudo-Intelligent Learning & Adaptation Technique

- Knowledge Building

  The outcome of each analysis stage builds the knowledge an agent operating on a mobile medical device has about an adaptive server within its environment. In effect, the information accumulated through the knowledge building process reflects the agent's overall opinion regarding the bid reliability of adaptive servers with whom it has interacted.

- Knowledge Acquisition

  The stages of *Performance Analysis* and *Knowledge Building* constitute the learning process of the agent operating on the mobile medical device, whereas the stages of *Knowledge Acquisition* and *Informed Action* formulate the adaptation process of the agent. Adaptation is focused upon utilising the knowledge gathered within the learning process to determine the most beneficial option in terms of adaptive server utilisation. Knowledge acquisition occurs when an agent operating on a mobile medical device is deciding upon the adaptive server to which it will offload a computational task. The stage is specifically concerned with obtaining records from a repository in relation to adaptive server bid accuracy.

- Informed Action

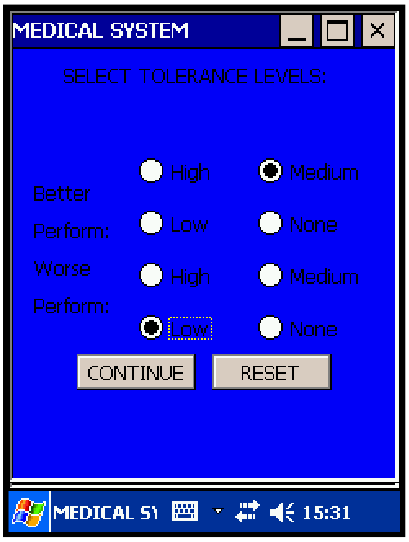
  This stage uses the records returned from the *Knowledge Acquisition* process to alter incoming bids from adaptive servers to service a computational request. This informed modification of adaptive server bids reflects the reliability of their previous bid accuracy. Bid adjustment can take the form of an enhancement or alternatively a downgrading of the bid.

Cause of Poor Bid Estimation	**Description**
Intermittent Wireless Connection	An intermittent wireless connection is an external factor outside of an adaptive server's control. It can hinder the ability of a server to return the results of a computational request to a handheld medical device. This delay can affect the bid accuracy of an adaptive server.
Arrival of Higher Priority Tasks	Learning and adaptation characteristics can be incorporated into participating agents within the context-based negotiation protocol outlined in Section 5.2.2. The protocol encourages re-organisation of an adaptive server's queue of jobs upon the arrival of tasks with high priority. This can result in an adaptive server taking longer to service a task than originally calculated within their bid estimate.
Under-estimation of Computational Ability	An adaptive server may under-estimate its computational processing power. This can lead to service time for a computational task taking less than the time originally proposed within a bid estimate.

Table 7.2: Potential Reasons for Poor Bid Estimation by Adaptive Servers

The *Performance Analysis* stage contained within the learning and adaptation negotiation protocol dictates the *Mobile Device Manager* agent's satisfaction or dissatisfaction with the performance of an adaptive server. Their degree of leniency is pre-determined by an administrator through a graphical user interface upon device initialisation.

This interface enables an agent operating on a mobile medical device to determine the degree of leniency to award to adaptive servers within the *Performance Analysis* stage. Their degree of leniency is pre-determined by an administrator through a graphical user interface upon device initialisation. A screenshot of this visual interface is shown in Figure 7.2.



MEDICAL SYSTEM

SELECT TOLERANCE LEVELS:

Better Perform: ○ High ● Medium ○ Low ○ None

Worse Perform: ○ High ○ Medium ● Low ○ None

Figure 7.2: Tolerance Levels Screenshot within Learning and Adaptation Negotiation Protocol

The tolerance levels dictate the cut-off point upon which the agent will begin imposing penalties or credits for an inaccurate bid. The percentage values of the threshold levels are shown in Table 7.3.

The percentage rating associated with each tolerance level determines the time differential acceptable by an agent in relation to bid time versus actual computational task service time.

When the actual service time violates a tolerance level, a penalty or a credit is applied to the *bid adjustment weighting* as shown in a pseudo-code extract presented in Figure 7.3. This *bid adjustment weighting* is utilised to modify an adaptive server's computational service bid to accurately reflect their previous record of performance.

Tolerance Level	**Percentage Rating** **(percentage a bid time can be outside actual time to service a computational task)**
High	20%
Medium	10%
Low	5%
None	Not Applicable

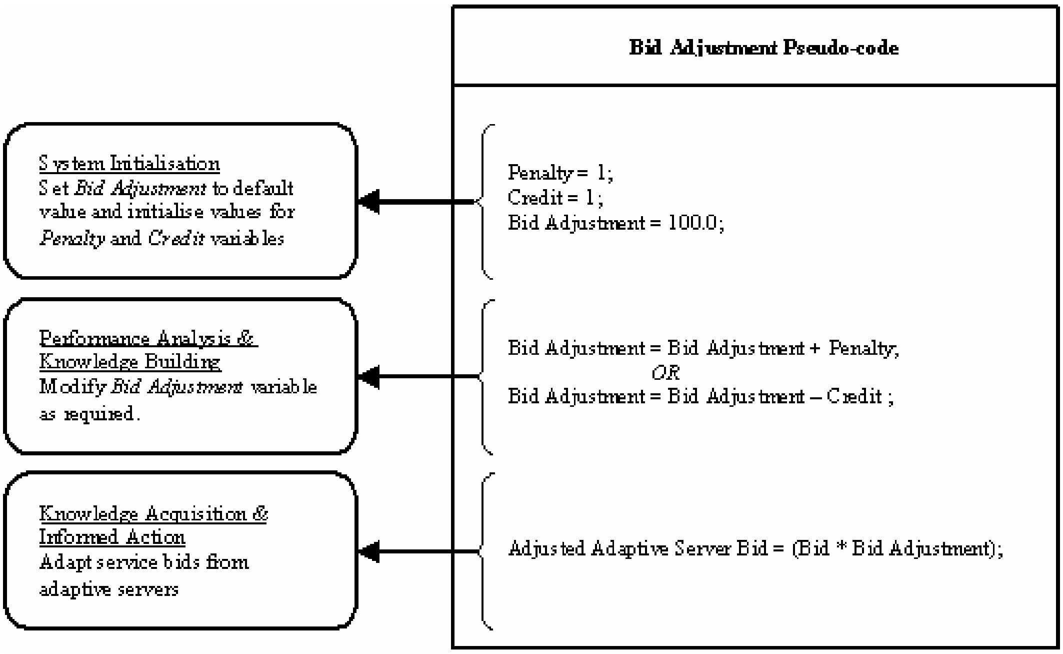
Table 7.3: Tolerance Level Percentage Ratings

7.2.1 Evaluation

A medical-based experimental prototype was constructed to evaluate the performance of the proposed context-aware negotiation protocol. The prototype enables a physician to retrieve patient scans that have been image processed in real-time by an adaptive server.

7.2.2 Test Case Environment

The test case environment utilises the same physical framework presented in Section 6.2.1.2. This presents a network environment consisting of a JADE-LEAP agent platform executing on a Dell Axim PDA. The handheld device communicates over a Wi-Fi network to an agent platform that comprises of four adaptive servers executing within agent containers on a high-end Pentium PC.


Bid Adjustment Pseudo-code
System Initialisation
Set *Bid Adjustment* to default value and initialise values for *Penalty* and *Credit* variables
Performance Analysis & Knowledge Building
Modify *Bid Adjustment* variable as required.
Knowledge Acquisition & Informed Action
Adapt service bids from adaptive servers

```
Penalty = 1;
Credit = 1;
Bid Adjustment = 100.0;

Bid Adjustment = Bid Adjustment + Penalty;
                 OR
Bid Adjustment = Bid Adjustment – Credit ;

Adjusted Adaptive Server Bid = (Bid * Bid Adjustment);
```

Figure 7.3: Bid Adjustment Pseudo-Code

The additional reasoning required for successful deployment of the learning and adaptation negotiation protocol is encapsulated within the participating agents that comprise the middleware within the distributed framework.

7.2.3 Performance Results

Test cases were conceived to examine the effectiveness of the proposed protocol in enabling a handheld medical device recognise and learn from misleading adaptive server bids. The learning and adaptation negotiation protocol enhances the quality of service experienced by a handheld device user by enabling an informed decision-making process regarding adaptive server utilisation.

A test case simulation examined the changes applied to a *bid adjustment weighting* by the negotiation protocol. These modifications were derived through simulated interactions between a handheld device and adaptive servers of varying reliability. The *threshold level* acceptable by the agent operating on the handheld device within simulation testing was set at *low* for an adaptive server delivering both better and worse performance times. A *low threshold level* has an associated percentage rating of 5%.

Three separate adaptive server reliability scenarios were conceived and these are shown in Figure 7.4. The Java-based simulation environment employed these scenarios to model adaptive servers of varying reliability.

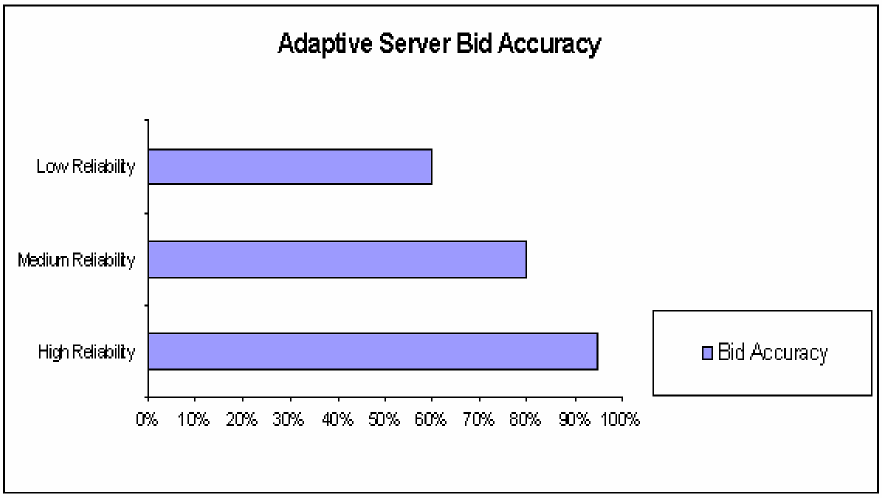


Figure 7.4: Adaptive Server Reliability Scenarios

The bid accuracy of the adaptive server scenarios modelled was as follows:

- Low Reliability Adaptive Server
  This server has a 40% chance of delivering an inaccurate bid to a handheld device.

- Medium Reliability Adaptive Server
  This server has an 80% chance of providing performance in accordance with its original bid to a handheld medical device.

- High Reliability Adaptive Server
  This adaptive server is the least likely within the three proposed scenarios to deliver an inaccurate bid. There is a 5% chance of a bid from this adaptive server proving inaccurate in comparison with actual performance levels.

The above adaptive server scenarios were applied within a simulated environment whereby each server was awarded a hundred computational tasks from a handheld medical device. The changes applied by the portable computer to its *bid adjustment weighting* variable were monitored. The results obtained from this simulated testing are shown in Figure 7.5. These results are applicable to a device with a low tolerance level for adaptive servers delivering both better and worse performance.

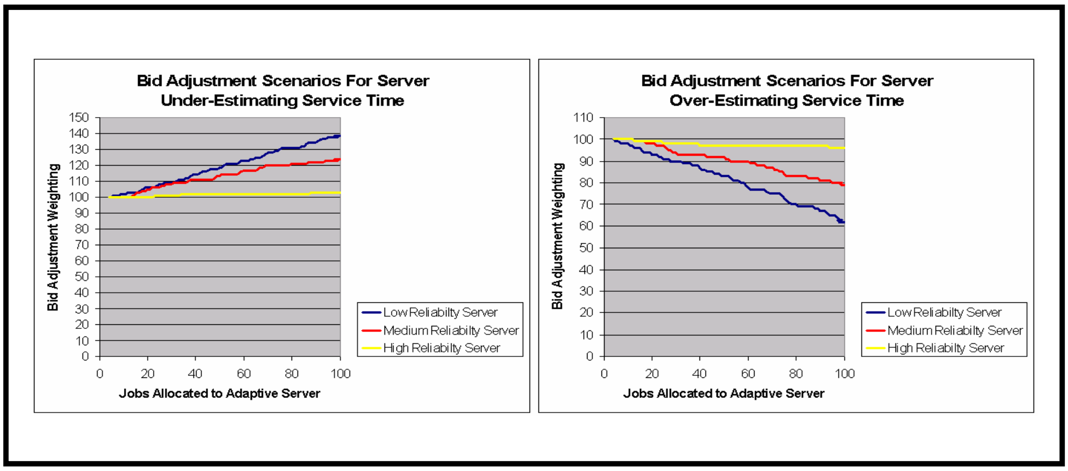


Figure 7.5: Bid Adjustment Scenarios for Adaptive Server Scenarios Under-Estimating & Over-Estimating Bids

The simulation environment models actual performance levels of an adaptive server by applying the following steps:

1. A random number was generated to determine if the server has delivered an inaccurate bid. The determinant applied was the reliability rating of the server as highlighted within Figure 7.4.
2. If an adaptive server was adjudged to have delivered an in-accurate bid then an additional random number was generated to determine if the performance of the adaptive server violated the *tolerance level* of the handheld device.
3. Random number generation was achieved through applying the Monte Carlo technique.

Simulation results shown in Figure 7.5 highlight the ability of a handheld device to learn about adaptive server reliability through its interactions with the servers. The modification made to the *bid adjustment weighting* reflects the adaptive measures undertaken by the portable device in responding to this learning experience.

The simulated scenario of a low reliability adaptive server that under-estimates service time shows it to be heavily penalised by the agent operating on the mobile device. Its *bid adjustment weighting* is shown to be 140 after a hundred computational offloads by the portable computer to this adaptive server. This represents a 40% increase by the handheld device on any future bid received from the adaptive server.

7.2.4 Future Research through Expansion of Learning Agents

The learning and adaptation protocol could be further expanded by increasing the number of learning agents within the distributed framework. Currently, learning and adaptation is constrained to operate within the handheld medical device. These characteristics could also be introduced to agents executing on adaptive servers. This would be especially beneficial within a network environment which combined the learning and adaptation protocol with the context-based negotiation strategy presented in Section 6.2.2.

Such a combined negotiation protocol would compromise the ability of an adaptive server to make accurate calculations as to computational task service time. The arrival of higher priority tasks as outlined in Table 7.2 could lead to an adaptive server making inaccurate computational task bids. Learning and adaptation abilities would enable agents operating on adaptive servers to make educated estimates as to the arrival rate of higher priority tasks. This would result in the calculation of a more accurate bid in response to a call for proposals from a handheld device.

7.3 Collective Past-Experience Learning Strategy

An agent operating on a handheld device can learn through interaction about adaptive servers within its environment and can use this knowledge to enhance quality of service for its owner. These learning and adaptation characteristics are integral components of the negotiation protocol presented in Section 7.2. The protocol enables agents executing on handheld devices to make informed decisions regarding adaptive server utilisation.

An extension of this protocol is to collect, correlate and share the knowledge of every agent on a handheld device within the network [O' Sullivan, 06c]. This collective past-experience learning strategy focuses strongly on the collaborative nature of multi-agent systems.

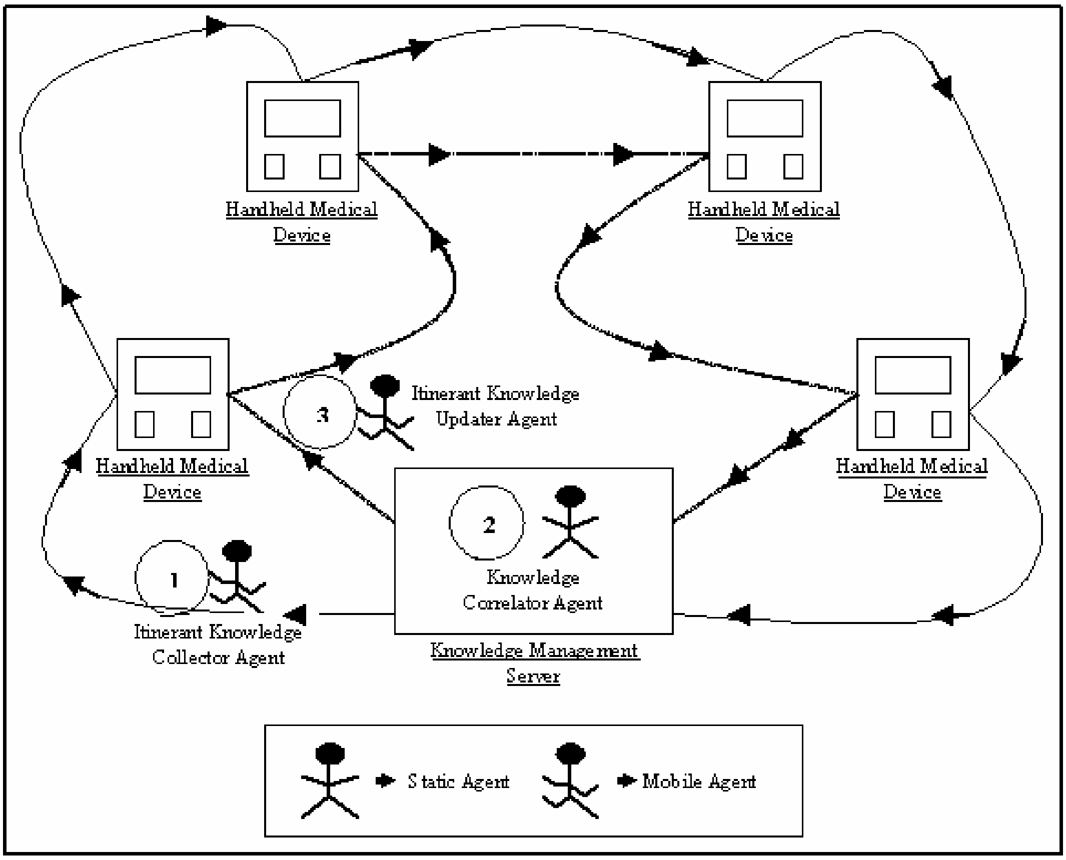
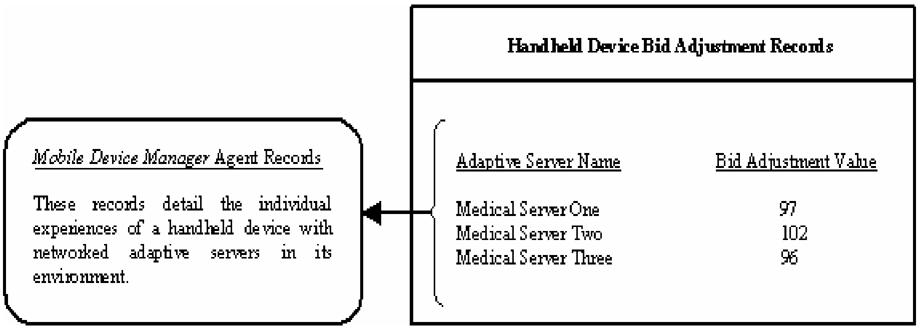

Handheld Medical Device
Handheld Medical Device
Handheld Medical Device
Handheld Medical Device
3
Itinerant Knowledge Updater Agent
2
Knowledge Correlator Agent
1
Itinerant Knowledge Collector Agent
Knowledge Management Server
Static Agent
Mobile Agent

Figure 7.6: Collective Past-Experience Learning Strategy

Strong collaboration among agents allows for both efficient pooling of knowledge resources and for effective dissemination of accumulated experiences. The collective learning strategy improves the knowledge base of each negotiating agent on a handheld device. This enhances their decision-making ability with regard to adaptive server utilisation. The collective past-experience learning strategy consists of three distinct phases which are shown through numbering in Figure 7.6 and can be described as follows:

- Itinerant Knowledge Collector Agent Phase

  This phase witnesses the traversal of all handheld devices by a mobile agent. The *Itinerant Knowledge Collector* agent moves between portable computers gathering their records of past-experience with adaptive servers. An example outlining the format of these past-experiences is shown in Figure 7.7. This shows a *Mobile Device Manager* agent documents its offloading experiences with adaptive servers according to an overall *bid adjustment* value. This *bid adjustment weighting* is utilised to modify an adaptive server's computational service bid to accurately reflect their previous record of performance.


Handheld Device Bid Adjustment Records
Mobile Device Manager Agent Records
These records detail the individual experiences of a handheld device with networked adaptive servers in its environment.

Adaptive Server Name	Bid Adjustment Value
Medical Server One	97
Medical Server Two	102
Medical Server Three	96

Figure 7.7: Handheld Device Bid Adjustment Records Example

The *Itinerant Knowledge Collector* agent commences its journey around the handheld medical devices from the *knowledge management* server. This server is the central point of control for executing the processes within the collective past-experience strategy. The *Itinerant Knowledge Collector* agent accesses a repository within this server which details the locations of portable computers within the network environment to which it will travel.

- Knowledge Correlator Agent Phase

  This phase observes the filtration of accumulated knowledge obtained by the *Itinerant Knowledge Collector* agent. The agent responsible for determining meaningful analysis from gathered data is the *Knowledge Correlator* agent. This is a static agent which resides within the *knowledge management* server. The *Knowledge Correlator* is the agent responsible for controlling all phases of the collective past-experience learning strategy. It initiates the movement of both the *Itinerant Knowledge Collector* and the *Itinerant Knowledge Updater* agents amongst the handheld medical devices. The formula devised to obtain an overall appraisal of handheld device impressions for each adaptive server within their environment is shown below:

$$\text{Bid Adjustment Appraisal} = \frac{\text{Bid Adjustment}^1 + \text{Bid Adjustment}^2 + \ldots\ldots + \text{Bid Adjustment}^N}{N}$$

Where N = Total Number of Handheld Devices within Network Environment

  This formula simply calculates the mean *bid adjustment* value for each adaptive server. This generates a clearer picture of individual adaptive server performance as the value calculated reflects opinions gathered from all handheld devices. There is further scope for extending the complexity of this appraisal process which is discussed in Section 7.3.4.

- Itinerant Knowledge Updater Phase

  This phase witnesses the dissemination of correlated data amongst all handheld devices within the network environment. The *Itinerant Knowledge Updater* agent moves to each portable computer updating their repository of *bid adjustment* values. This mobility of the agent facilitates local interaction with handheld device resources and this helps reduce network bandwidth usage. Mobility is also a highly suitable mechanism to combat intermittent wireless connections. The *Itinerant Knowledge Updater* agent will always re-attempt to move to the next destination on its path and is capable of operating with sporadic network connectivity.

The agent architectural roles of *Itinerant Knowledge Collector*, *Knowledge Correlator*, and *Itinerant Knowledge Updater* within the collective past-experience learning strategy were developed using an agent-oriented analysis and design methodology [Wooldridge, 99].

7.3.1 Evaluation

A system framework was constructed to effectively evaluate the potential of the collective past-experience learning strategy. This evaluation prototype extended the medical-based system outlined in Section 7.2.1 which was utilised to deploy the learning and adaptation negotiation protocol.

The collective learning strategy is initiated by an administrator through a graphical user interface. This input screen executes within the *knowledge management* server and is displayed upon initialisation of the collective learning strategy application. Activation of the *Itinerant Knowledge Collector* agent occurs through an administrator-driven manual process.

The collective learning strategy would be an automated process within a real-world setting. The interval between complete cycles of the process would be administrator defined. The learning strategy would be executed during periods of low network activity as part of a bandwidth and handheld device resource conservation policy.

7.3.2 Test Case Environment

The test case environment comprises of the *knowledge management* server executing on a Pentium PC. Logic implementing the collective learning strategy operates within a JADE agent platform on this server. The high-performance desktop computer is connected through a USB[70] cable to a Wi-Fi beacon. This establishes a private wired/wireless network.

[70] Universal Serial Bus

A Dell Inspiron Mobile Notebook communicates over the wireless portion of this network with the *knowledge management* server [Dell, 05b]. This notebook executes four separate JADE agent containers concurrently. Each individual container emulates an individual handheld device within the test case environment.

7.3.3 Performance Results

A test was conceived to examine the effectiveness of collaboration amongst agents within the collective past-experience learning strategy. This test was executed within two simulated real-world network scenarios. The potential changes applied to *bid adjustment* records within a handheld device by an *Itinerant Knowledge Updater* agent operating through the collective learning strategy were examined.

A purpose-built Java simulator was constructed and this used data obtained from the test-case environment to achieve reliable analysis. The network scenarios comprised of three adaptive servers with varying degrees of bid estimation accuracy. These simulated adaptive servers are drawn from the reliability scenarios presented in Figure 7.4 and included a *low reliability*, a *medium reliability*, and a *high reliability* server. The simulation modelled servers which under-estimated computational task service time with these reliability settings.

Four handheld devices were utilised to offload six hundred computational tasks amongst the three adaptive servers (i.e. approximately one hundred and fifty tasks originating from each portable computer). These tasks were distributed to the adaptive servers equally. As each task was assigned to an adaptive server, an additional random number was generated using the Monte Carlo random number generation technique which determined if the server delivered an inaccurate bid. The determinant applied was the reliability rating of the adaptive server.

The *tolerance level* setting for handheld devices within the simulation was disregarded so as to simplify the test case scenario. If an adaptive server was adjudged to have delivered an in-accurate bid then modifications were applied to its *bid adjustment weighting*.

The performance results of the initial simulation are presented in Figure 7.8. These bar charts show the *bid adjustment weightings* of a handheld device before and after the execution of the collective past-experience learning strategy.

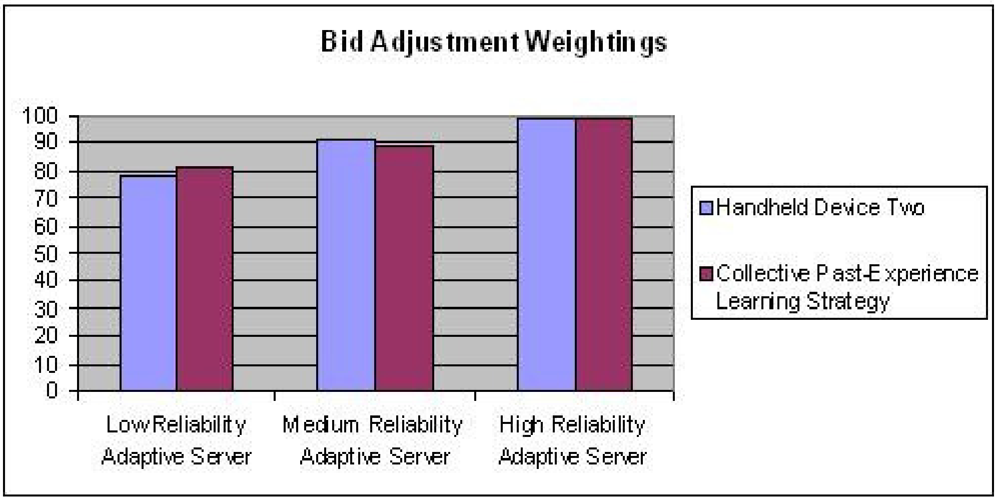


Figure 7.8: Initial Simulation Performance Results

The results clearly highlight the learning experiences of the individual mobile device and of the wider handheld community to be very similar. This is known as the *reinforcement* characteristic of the collective learning strategy. Essentially, beliefs are reinforced within a handheld device community for adaptive servers with consistently reliable or unreliable bid estimates. The reinforcement characteristic is synopsised within the graphic presented in Figure 7.9.

This is an important attribute of the collective learning strategy as it ensures the protocol does not unduly alter widely held beliefs within the agent community.

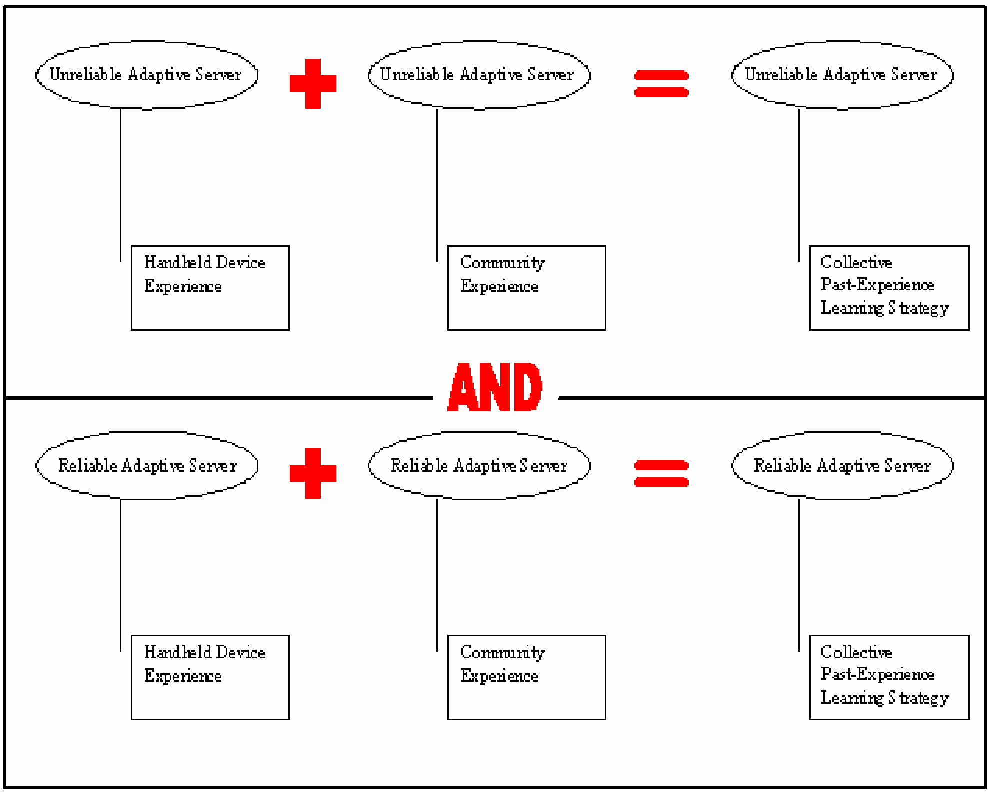

Unreliable Adaptive Server
Handheld Device Experience
Unreliable Adaptive Server
Community Experience
Unreliable Adaptive Server
Collective Past-Experience Learning Strategy
AND
Reliable Adaptive Server
Handheld Device Experience
Reliable Adaptive Server
Community Experience
Reliable Adaptive Server
Collective Past-Experience Learning Strategy

Figure 7.9: Reinforcement Characteristic of Collective Past-Experience Learning Strategy

The second network simulation sought to alter the learning experience of an individual mobile device in comparison with the wider handheld community and to monitor the effect of the collective learning strategy on this device's beliefs. The environment settings for the simulator model were modified such that any computational tasks offloaded to the high reliability server by the second handheld device were dealt with in a low reliability manner. The performance results for this simulation are presented in Figure 7.10.

The results highlight another generalised recurring characteristic of the collective learning strategy. Essentially, a belief held by a handheld device about the bid reliability of an adaptive server may be forced to change if it is contrary to the wider held belief within the agent community. This is apparent within the second simulation performance results graph in relation to the substantial belief change experienced by the individual handheld device. Its *bid adjustment weighting* is subject to an alteration increase of 19% which drastically modifies its view of the bid reliability of the third server.

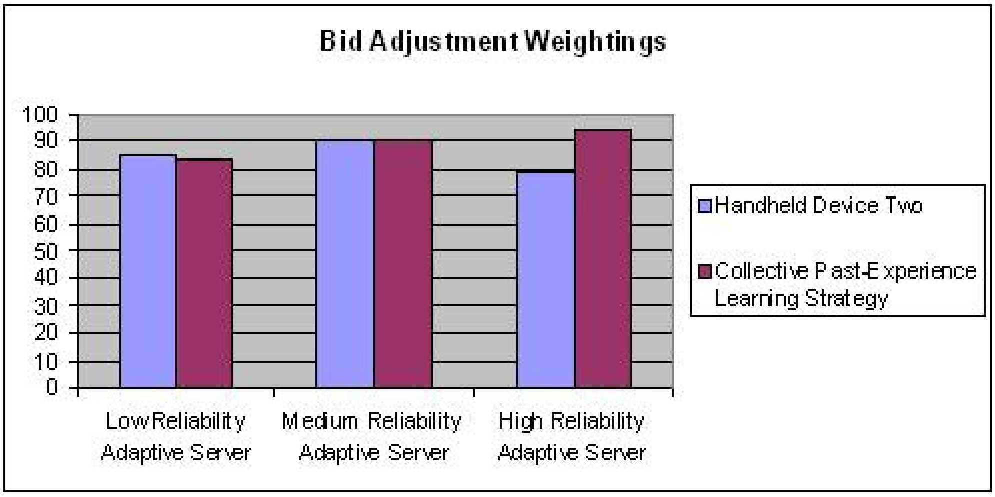


Figure 7.10: Second Simulation Performance Results

The second simulation environment is clearly staged such that a belief of an individual handheld is contrary to the wider community belief. However, the point of interest is the capability of the collective past-experience learning strategy to depict a more accurate picture of adaptive server reliability for all handheld devices.

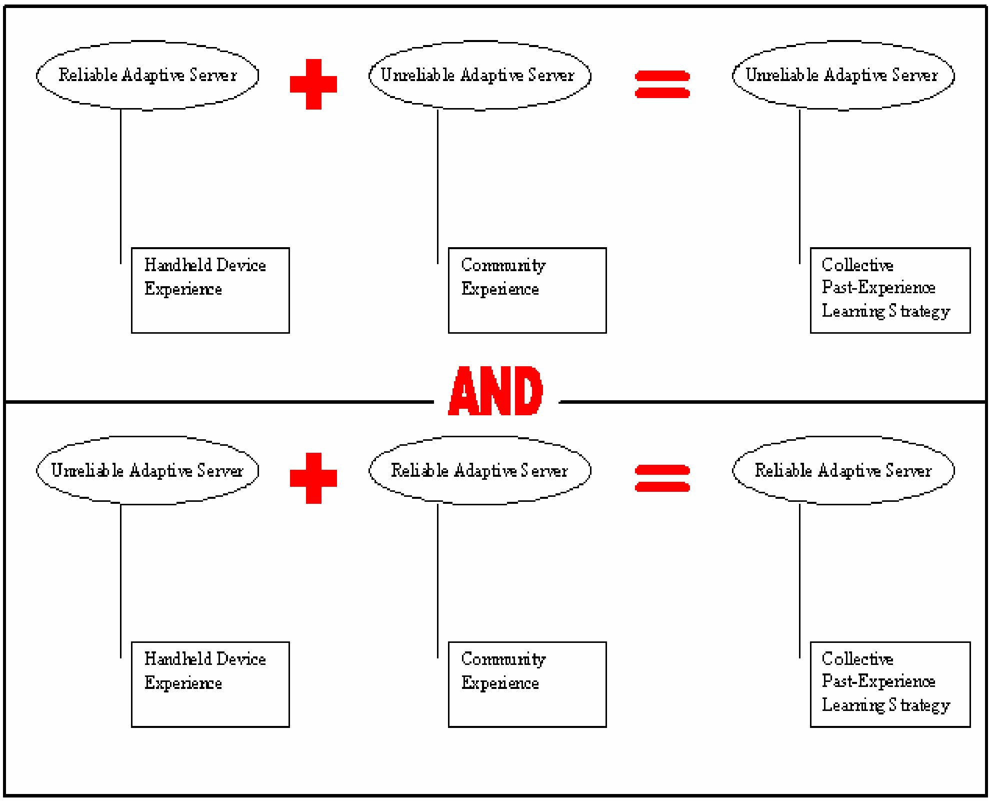

Reliable Adaptive Server
Unreliable Adaptive Server
Unreliable Adaptive Server
Handheld Device Experience
Community Experience
Collective Past-Experience Learning Strategy
AND
Unreliable Adaptive Server
Reliable Adaptive Server
Reliable Adaptive Server
Handheld Device Experience
Community Experience
Collective Past-Experience Learning Strategy

Figure 7.11: Enlightenment Characteristic of Collective Past-Experience Learning Strategy

This enlightenment characteristic is synopsised within the diagram presented in Figure 7.11. The attribute highlights a beneficial characteristic of the collective learning strategy as it shows collaborative sharing of knowledge will influence individually held beliefs. This is preferential as the wider community experience is a more dependable indicator of the actual bid reliability of adaptive servers.

7.3.4 Future Research

Future research would introduce finer granularity to the records held by a handheld device in relation to adaptive server bid reliability. Further data concerning the environmental conditions experienced by a handheld device when offloading a computational task would enrich the *knowledge correlation* phase of the collective past-experience learning strategy.

For example, an intermittent wireless connection is an external factor outside of an adaptive server's control. It can hinder the ability of a server to return the results of a computational request to a handheld medical device. This delay can affect the bid accuracy of an adaptive server.

However, the wireless connection may only operate erratically within a small geographic area of the overall hospital environment. Being aware of the physical geographic locations associated with an offloaded computational request would ensure the *knowledge correlation* phase can take into account mitigating factors when setting the *bid adjustment weighting* of an adaptive server.

It should be noted that the primary disadvantage of enhancing the collective past-experience learning strategy is the additional demands placed upon system resources throughout the network. The demands relate to recording, gathering, correlating and disseminating supplementary data.

CHAPTER 8

Thesis Evaluation and Conclusions

8.1 Introduction

Pervasive handheld computing systems are multi-functional mobile devices that are capable of executing a broad range of compute-intensive applications. This thesis outlines a two-pronged methodology enabling *pervasive handheld computing systems* to meet their performance and versatility requirements.

The proposed methodology is a fusion of two independent whilst complementary concepts. The initial step employs reconfigurable technology to enhance the physical hardware resources within the environment of a portable computer. Reconfigurable hardware can dynamically increase the system functionality and versatility of a handheld device without major loss in performance. The second step of the methodology introduces agent-based middleware protocols to support mobile devices to effectively manage and utilise available reconfigurable hardware resources within their environment.

Effectively, this two-pronged methodology places an equal emphasis upon the need for high-performance hardware resources within handheld device environments and upon a sophisticated middleware to enable effective management of these aforementioned resources. The thesis argues that this methodology constitutes a coherent approach to meeting the performance and versatility objectives of *pervasive handheld computing systems*.

8.2 Contributions and Results

This dissertation makes a number of research contributions to the field of *pervasive handheld computing systems*. Each research contribution is novel in nature and has been performance evaluated. These contributions can be summarised as follows:

- A mixed agent-object design technique advocating the use of both agents and objects within agent-oriented systems is presented within Section 4.8. This design strategy helps produce agent-oriented designs which achieve a healthy balance between agents and objects.

- An intelligent medical-oriented framework denoted as CAMMD is outlined within Chapter Five. CAMMD provides handheld medical devices with a support infrastructure capable of capturing, communicating and interpreting real-time contextual information. The CAMMD framework is shown to enhance the usability and portability of mobile medical devices. The methodology embodied within the architecture also helps overcome handheld device and network limitation issues.

- An agent-based negotiation protocol enabling a handheld device to efficiently offload reconfigurable hardware-software based computations to neighbouring adaptive servers is presented within Section 6.2.1. The performance of the protocol is evaluated through a range of test case scenarios. These scenarios compare performance with results gathered from a traditional client-server framework implementation. The protocol is shown to deliver vast service time improvements to users of handheld devices within networks of unevenly distributed loads.

- A context-based negotiation strategy facilitating intelligent utilisation of networked reconfigurable resources by portable computers is outlined within Section 6.2.2. This protocol employs contextual information to better inform the decision-making process of agents operating within adaptive servers with regard to resource allocation. Test cases were applied to an experimental prototype to evaluate the ability of the protocol to recognise the urgency of a computational request. The context-based negotiation protocol is shown to deliver better performance than the agent-based negotiation strategy for medium, high, and urgent priority-based requests.

- A push-based configuration management strategy for the coherent distribution of reconfigurable-hardware based application solutions to handheld devices is presented within Section 6.3.2. The mobile agent concept is employed to completely encapsulate and transport all necessary reconfigurable hardware and software code representing an application solution. The configuration management protocol was developed using agent-based design patterns and an agent-oriented analysis and design methodology known as Gaia.

- A protocol facilitating context-aware hardware reconfiguration for handheld devices is outlined within Section 6.3.3. This protocol seeks to enhance the utilisation of reconfigurable resources contained within a portable computer. Essentially, the protocol propagates reconfigurable hardware-based applications to a handheld device depending upon the contextual environment of its user. This enables a portable computer to have its hardware configuration dynamically tailored according to the usage (i.e. application) requirements of a user. An implementation framework was developed which showed the context-aware reconfiguration protocol improved user interaction by helping to avoid information overload. This prototype system also highlighted the ability of the protocol to effectively utilise portable computer storage helping to improve device portability.

- A learning and adaptation negotiation protocol enabling handheld devices to more effectively exploit adaptive server technology is presented within Section 7.2. Learning and adaptation are agent characteristics which can enhance the decision-making process regarding networked reconfigurable hardware utilisation on a portable computer. A test case environment was constructed to examine handheld medical devices employing the protocol. This enabled the device to recognise and learn from misleading adaptive server bids. Simulation results verified the ability of the protocol to enable a handheld device to learn about the reliability of adaptive servers within its network environment. Simulation results also showed the protocol allowed the device to adapt its behaviour accordingly.

- A collective past-experience learning strategy improving the knowledge base of negotiating agents within a distributed environment is outlined within Section 7.3. This protocol collects, correlates, and shares adaptive server knowledge of every negotiating agent on a handheld device within a network. The collective past-experience learning strategy focuses strongly upon the collaborative nature of multi-agent systems. Strong collaboration amongst agents allows for both efficient pooling of knowledge resources and for effective dissemination of accumulated experiences. A system framework was constructed to effectively evaluate the potential of the protocol. Test case results were distinguished according to the effect of the protocol upon the knowledge base of an individual handheld device. These belief changes are categorised as *reinforcement* and *enlightenment* characteristics and are detailed further within Section 7.3.3.

8.3 Future Work

The field of *pervasive handheld computing systems* is relatively young and requires much further research and development. The following topics have been identified within the course of this work as areas which would benefit the field of *pervasive handheld computing systems*:

- Partitioning and scheduling constraints become increasingly difficult to obey as the degree of reconfiguration flexibility increases. This is primarily attributable to the issues of configuration latency and partial reconfiguration. The partitioning and scheduling of tasks within the prototypes developed to demonstrate *pervasive handheld computing systems* were pre-determined manually with the aid of system-level designs. Future research will expand upon this approach to incorporate automated partitioning and scheduling techniques.

- The context-based negotiation protocol outlined within Section 6.2.2 would benefit by interpreting additional contextual data elements within the handheld device environment. An example scenario of an enhanced context-based strategy operating within a telemedicine environment is detailed further within Section 6.2.2.3.4. This scenario employs real-time sensors physically located upon a patient to better inform the decision-making process of agents. Essentially, an agent operating on an adaptive server would determine the priority of a medical practitioner's computational request according to the location of the handheld device and the current state of the patient associated with the request.

- The push-based configuration management strategy presented within Section 6.3.2 would be enhanced by introducing security-based concepts to the framework. An overview of a security approach which could be employed is detailed within Section 6.3.2.6.

- The learning and adaptation negotiation protocol introduced within Section 7.2 could be enhanced by facilitating those agents operating within adaptive servers with learning and adaptation characteristics. This concept and the associated benefits are further expanded upon within Section 7.2.4.

- The collective past-experience learning strategy presented within Section 7.3 would benefit from the introduction of finer granularity in relation to adaptive server bid reliability. Essentially, additional data concerning the environmental conditions experienced by a handheld device when offloading a computational task would enrich the *knowledge correlation* phase of the collective past-experience learning strategy. The implications and benefits of this enhancement are detailed further within Section 7.3.4.

BIBLIOGRAPHY

[Abowd, 02] G. D. Abowd, E. D. Mynatt and T. Rodden, The Human Experience, In Proceedings of the IEEE Pervasive Computing Journal, vol. 1, no. 1, pp.48-57, (2002).

[Adario, 97] A. M. S. Adario, M. L. Cortes and N. J. Leite, An FPGA Implementation of a Neighbourhood Processor for Digital Image Applications, In Proceedings of the 10th Brazilian Symposium on Integrated Circuit Design, (1997).

[Adario, 99] A. M. S. Adario, E. L. Roehe, S. Bampi, Dynamically Reconfigurable Architecture for Image Processor Applications, In Proceedings of the ACM Design Automation Conference, (1999).

[Agarwal, 94] L. Agarwal, M. Wazlowski and S. Ghosh, An Asynchronous Approach to Efficient Execution of Programs on Adaptive Architectures Utilising FPGAs, In Proceedings of Second IEEE Workshop on FPGAs for Custom Computing Machines, pp. 101-110, (1994).

[AgentBuilder, 00] AgentBuilder: An integrated toolkit for constructing intelligent software agents, White Paper, (2000). http://www.agentbuilder.com.

[Aksoy, 98] D. Aksoy, et al, Research in Data Broadcast and Dissemination, In Proceedings of the 1st International Conference on Advanced Multimedia Content Processing, (1998).

[Albuquerque, 01] R. L. Albuquerque, et al, KSACI: A Handheld Device Infrastructure for Agents Communications, In Proceedings of the Eight International Workshop on Agent Theories, Architectures and Languages, (2001).

[Alexander, 79] C. Alexander, The Timeless Way of Building, Oxford University Press, New York, (1979).

[Altmann, 01] J. Altmann, F. Gruber, L. Klug, W. Stockner and E. Weippl, Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms, In Proceedings of the 2nd Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents Conference, (2001).

[Ancona, 01] M. Ancona, et al, Ward In Hand: Wireless Access to Clinical Records for Mobile Healthcare Professionals, In Proceedings of the 1st Annual Conference on Mobile and Wireless Healthcare Applications, (2001).

[Aridor, 98] Y. Aridor and D. B. Lange, Agent Design Patterns: Elements of Agent Application Design, In Proceedings of the Autonomous Agents Conference, (1998).

[Arnold, 99] K. Arnold, A. Wollrath, B. O' Sullivan, R. Scheifler, and J. Waldo, The Jini Specification, Addison-Wesley Publications, (1999).

[Ashenden, 96] P. J. Ashenden, The Designer's Guide to VHDL, Morgan Kaufmann Publishers, San Francisco, USA, (1996).

[Ashenden, 98] P. J. Ashenden, The Student's Guide to VHDL, Morgan Kaufmann Publishers, San Francisco, USA, (1998).

[Baldi, 97] M. Baldi, S. Gai and G. Picco, Exploiting Code Mobility in Decentralised and Flexible Network Management, In Proceedings of the Mobile Agents Conference, (1997).

[Bardram, 04] J. Bardram, Applications of Context-Aware Computing in Hospital Work – Examples and Design Principles, In Proceedings of the ACM Symposium on Applied Computing, (2004).

[Barton, 02] J. Barton, et al, Miniaturised Modular Wireless Sensor Networks, In Proceedings of the International Conference on Ubiquitous Computing, (2002).

[Belkin, 05] Wireless Router D-Link AirPlus DI-614 Datasheet available at http://www.belkin.com

[Bellifemine, 99] F. Bellifemine, A. Poggi, and G. Rimassa, JADE – A FIPA Compliant Agent Framework, In Proceedings of the International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems, pp. 97-108, (1999).

[Bellifemine, 00] F. Bellifemine, A. Poggi, G. Rimassa and P. Turci, An Object-Oriented Framework to Realise Agent Systems, In Proceedings of the Workshop From Objects to Agents, pp. 52-57, (2000).

[Bellifemine, 03] F. Bellifemine, G. Caire, A. Poggi and G. Rimassa, JADE A White Paper, In Proceedings of the TILAB Journal, (2003).

[Benedetti, 98] A. Benedetti and P. Perona, Real-Time 2-D Feature Detection on a Reconfigurable Computer, In Proceedings of the IEEE Conference on Computer Vision and Pattern Detection, (1998).

[Benitez, 99] D. Benitez and J. Cabrera, Reactive Computer Vision System with Reconfigurable Architecture, In Proceedings of the International Conference on Vision Systems, (1999).

[Bergenti, 01] F. Bergenti and A. Poggi, LEAP: A FIPA Platform for Handheld and Mobile Devices, In: Proceedings of the workshop on agent theories, architectures, and languages, (2001).

[Berger, 03] M. Berger, et al, Porting Agents to Small Mobile Devices: The Development of the Lightweight Extensible Agent Platform, In Proceedings of the TILAB Journal, (2003).

[Berghoff, 96] J. Berghoff, O. Drobnik, A. Lingnau and C. Monch, Agent-Based Configuration Management of Distributed Applications, In Proceedings of 3rd International Conference on Configurable Distributed Systems, (1996).

[BMA, 04] Quality & Outcomes Framework Guidance, British Medical Association, available at: http://www.bma.org.uk/ap.nsf/Content/QualityOutcomes, (2004).

[Booch, 98] G. Booch, J. Rambaugh and I. Jacobson, The Unified Modelling Language User Guide, Addision-Wesley, Reading, (1998).

[Burbeck, 04] K. Burbeck, D. Garpe and S. Nadjm-Tehrani, Scale-up and Performance Studies of Three Agent Platforms, In Proceedings of the International Workshop on Middleware Performance, (2004).

[Burmeister, 96] B. Burmeister, Models and methodology for agent-oriented analysis and design, In Proceedings of the Agent-oriented Programming and Distributed Systems Workshop, (1996).

[Burger, 97] D. Burger and J. Goodman, Billion-Transistor Architectures, In Proceedings of the Computer Journal, vol. 30, no. 9, pp.46-49, (1997).

[Buschmann, 95] F. Buschmann, The Master-Slave Pattern, Pattern Languages of Program Design, editors: O. J. Coplien and D. Schmidt, Addison Wesley, pp. 133-142, (1995).

[Buschmann, 96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, Pattern-Oriented Software Architecture: A System of Patterns, Wiley, Chichester, (1996).

[Caire, 02] G. Caire, N. Lhuillier, and G. Rimassa, A Communication Protocol for Agents on Handheld Devices, In Proceedings of Autonomous Agents and Multi-Agent Systems Conference, (2002).

[Capra, 02] L. Capra, Mobile Computing Middleware for Context-Aware Applications, In Proceedings of the International Conference on Software Engineering, (2002).

[Carabelea, 03] C. Carabelea, O. Boissier, and F. Ramparany, Benefits and Requirements of using Multi-Agent Systems on Smart Devices, In Proceedings of the 9th International Euro-Par Conference, pp. 1091-1098, (2003).

[Carabelea, 03b] C. Carabelea and O. Boissier, Multi-Agent Platforms on Smart Devices: Dream or Reality?, In Proceedings of the Smart Objects Conference, pp.126-129, (2003).

[Carzaniga, 97] A. Carzaniga, G. P. Picco and G. Vigna, Designing Distributed Applications with Mobile Code Paradigms, In Proceedings of the 19th International Conference on Software Engineering, (1997).

[Casselman, 02] S. Casselman and J. Schewel, Net Aware Bitstreams that Upgrade FPGA Hardware Remotely Over the Internet, In Proceedings of the SPIE Conference, vol. 4867, (2002).

[Celoxica, 05a] Celoxica RC200 Datasheet available at http://www.celoxica.com

[Celoxica, 05b] Celoxica Corporation, http://www.celoxica.com

[Chalmers, 98] D. Chalmers, Quality of Service in Mobile Environments, Masters Dissertation, Department of Computing, Imperial College, London, (1998).

[Chavez, 97] A. Chavez, A. Moukas and P. Maes, Challenger: A Multi-Agent System for Distributed Resource Allocation, In Proceedings of the First International Conference on Autonomous Agents, (1997).

[Chmiel, 04] K. Chmiel, et al, Testing the Efficiency of JADE Agent Platform, In Proceedings of the Third International Symposium on Parallel and Distributed Computing, pp. 49-56, (2004).

[Compton, 02] K. Compton, and S. Hauck, Reconfigurable Computing: A Survey of Systems and Software, In ACM Computing Surveys, vol. 34, no. 2, pp. 171-210, (2002).

[Cret, 02] O. Cret, Z. Baruch, and K. Pusztai, FPGAW: FPGA Configuration Over the Internet, In Proceedings of the

Third International Symposium on Mathematical and Computational Applications, (2002).

[Daggu, 04] R. V. Daggu, and V. Muthukumar, An Efficient Reconfigurable Architecture and Implementation of Edge Detection Algorithm using Handel-C, In Proceedings of the International Conference on Information Technology: Coding and Computing, (2004).

[Dave, 99] B. P. Dave, Crusade: Hardware/Software Co-Synthesis of Dynamically Reconfigurable Heterogeneous Real-Time Distributed Embedded Systems, In Proceedings of the Design, Automation and Test in Europe Conference, pp. 97-104, (1999).

[de Araujo Lima, 03] E. F. de Araujo Lima, P. D. de Lima Machado, J. C. A. de Figueiredo and F. R. Sampaio, Implementing Mobile Agent Design Patterns in the Jade Framework, In Proceedings of the TILAB Journal, (2003).

[Deepakumara, 01] J. Deepakumara, H. M. Heys, and R. Venkatesan, FPGA Implementation of MD5 Hash Algorithm, In Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, (2001).

[Dell, 05a] Dell Axim X3 Pocket PC Datasheet available at http://www.dell.com

[Dell, 05b] Dell Inspiron 510m Notebook Datasheet available at http://www.dell.com

[Della Mea, 01] V. Della Mea, Agents Acting and Moving in Healthcare Scenario: A New Paradigm for Telemedical Collaboration, In Proceedings of the IEEE Transactions on Information Technology in Biomedicine, (2001).

[Deugo, 99] D. L. Deugo, Communication as a Means to Differentiate Objects, In Proceedings of the Technology of Object-Oriented Languages & Systems Conference, (1999).

[Dick, 98] R. P. Dick and N. K. Jha, Cords: Hardware-Software Co-Synthesis of Reconfigurable Real-Time Distributed Embedded Systems, In Proceedings of the International Conference on Computer-Aided Design, pp. 62-68, (1998).

[D'Inverno, 97] M. D'Inverno et al, A Formal Specification of dMARS, In Intelligent Agents IV, editors: A. Rao, M. P. Singh and M. J. Wooldridge, vol. 1365, pp. 155-176, (1997).

[Doss, 99] C. C. Doss, and C. S. Clay, Automated Interface Generation for Remote Access to Adaptive Computing Resources, In Proceedings of the Military and Aerospace Programmable Logic Devices Conference, (1999).

[Draper, 02] B. Draper, et al, Implementing Image Applications on FPGAs, In Proceedings of the 16th International Conference on Pattern Recognition, (2002).

[Edmonds, 01] T. Edmonds, S. Hodges, and A. Hopper, Pervasive Adaptation for Mobile Computing, In Proceedings of the 15th International IEEE Information Networking Conference, (2001).

[Edwards, 05] S. A. Edwards, The Challenges of Hardware Synthesis from C-Like Languages, In Proceedings of the Design, Automation and Test in Europe Conference, pp. 66-67, (2005).

[Estrin, 63] G. Estrin, Parallel Processing in a Restructurable Computer System, In Proceedings of IEEE Transactions on Electronic Computers, (1963).

[FIPA, 05] The Foundation for Intelligent Physical Agents, http://www.fipa.org

[FIPA, 05b] FIPA Agent Communication Language Specification, available at http://www.fipa.org

[FIPA, 05c] FIPA Agent Management Specification, available at http://www.fipa.org

[Fleischmann, 99] J. Fleischmann, K. Buchenrieder, and R. Kress, Java Driven Codesign and Prototyping of Networked Embedded Systems, In Proceedings of the Design Automation Conference, (1999).

[Fleischmann, 99b] J. Fleischmann and K. Buchenrieder, Prototyping Networked Embedded Systems, In Proceedings of the IEEE Computer Journal, vol. 32, no. 2, pp. 116-119 (1999).

[Forgy, 82] C. Forgy, Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem, In Proceedings of the Artificial Intelligence Journal, vol. 19, pp. 17-37, (1982).

[Francis, 92] R. J. Francis, A Tutorial on Logic Synthesis for Lookup-Table Based FPGAs, In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 40- 47, (1992).

[Friedman-Hill, 03] E. Friedman-Hill, Jess in Action: Rule-Based Systems in Java, Manning Publications, (2003).

[Fuggetta, 98] A. Fuggetta, G. Pietro Picco and G. Vigna, Understanding Code Mobility, In Proceedings of the IEEE Transactions on Software Engineering, vol. 24, no. 5, (1998).

[Gaddah, 03] A. Gaddah and T. Kunz, A Survey of Middleware Paradigms for Mobile Computing, Technical Report SCE-03-16, Department of Systems and Computing Engineering, Carleton University, (2003).

[Gaj, 03] K. Gaj, et al, Effective Utilization and Reconfiguration of Distributed Hardware Resources using Job Management Systems, In Proceedings of the International Parallel and Distributed Processing Symposium, (2003).

[Genesereth, 94] M. R. Genesereth, and S. P. Ketchpel, Software Agents, In Communications of the ACM, vol. 37, no. 7, pp. 48-53, (1994).

[Gamma, 95] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns, Addison-Wesley, Reading, (1995).

[Giarratano, 98] J. Giarratano and G. D. Riley, Expert Systems: Principles and Programming, Third Edition, PWS Publishing Company, Boston, (1998).

[Gottlieb, 03] D. B. Gottlieb and N. P. Carter, Microprocessor Interfacing Laboratory, In Proceedings of the International Conference on Microelectronic Systems Education, (2003).

[Gray, 00] R. Gray, D. Kotz, G. Cybenko and D. Rus, Mobile agents: Motivations and state-of-the-art systems, Research Report TR2000-365, Dartmouth College, USA, (2000).

[Gruber, 93] T. Gruber, A Translation Approach to Portable Ontology Specifications, In Proceedings of the Knowledge Acquisition Journal, vol. 5, no. 2, pp. 199-220, (1993).

[Guccione, 99] S. Guccione, D. Levi, and P. Sundararajan, JBits: Java Based Interface for Reconfigurable Computing, In Proceedings of 2nd Annual Military and Aerospace Applications of Programmable Devices and Technologies Conference, (1999).

[Guccione, 02] S. Guccione, D. Verkest, and I. Bolsens, Design Technology for Networked Reconfigurable FPGA Platforms, In Proceedings of Design, Automation and Test in Europe Conference, pp. 994-997, (2002).

[Ha, 99] Y. Ha, et al, A Scalable Architecture to Support Networked Reconfiguration, In Proceedings of the IEEE Program for Research on Integrated Systems and Circuits, pp. 677-683, (1999).

[Ha, 01] Y. Ha, et al, Virtual Java / FPGA Interface for Networked Reconfiguration, In Proceedings of the Asia and South Pacific Design Automation Conference, pp. 558-563, (2001).

[Ha, 02] Y. Ha, et al, Building a Virtual Framework for Networked Reconfigurable Hardware and Software Objects, In Proceedings of the Journal of Supercomputing, vol. 21, no.2, pp.131-144 (2002).

[Hall, 98] S. R. Hall, D. Heimbiger and L. A. Wolf, A Cooperative Approach to Support Software Deployment Using the Software Dock, In Proceedings of the International Conference on Software Engineering, pp. 174-183, (1998).

[Handel-C, 05] Handel-C White Paper available at http://www.celoxica.com

[Harr, 00] R. Harr, The Nimple Compiler for Agile Hardware: A Research Platform, In Proceedings of the 13th International Symposium on System Synthesis, (2000).

[Hartenstein, 95] R. W. Hartenstein, et al, A Reconfigurable Machine for Applications in Image and Video Compression, In Proceedings of the Conference on Compression Technologies and Standards for Image and Video Compression, (1995).

[Hauck, 98] S. Hauck, The Roles of FPGAs in Reprogrammable Systems, In Proceedings of the IEEE, vol. 86, no. 4, pp. 615-638, (1998).

[Hauser, 98] J. R. Hauser and J. Wawrzynek, Garp: A MIPS Processor with a Reconfigurable Coprocessor, In Proceedings of the IEEE Symposium of Field-Programmable Custom Computing Machines, pp. 12-21, (1998).

[Hedberg, 03] H. Hedberg, et al, Teaching Digital HW-Design by Implementing a Complete MP3 Decoder, In Proceedings of the International Conference on Microelectronic Systems Education, (2003).

[Helin, 03] H. Helin, Supporting Nomadic Agent-based Applications in the FIPA Agent Architecture, PhD Thesis, Department of Computer Science, University of Helsinki, Finland, (2003).

[Hennessy, 96] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kauffman Publishers, San Francisco, (1996).

[Hightower, 04] J. Hightower and G. Borriello, Particle Filters for Location Estimation in Ubiquitous Computing: A Case Study, In Proceedings of the International Conference on Ubiquitous Computing, (2004).

[Huang, 95] J. Huang, N. R. Jennings, and J. Fox, An Agent-based Approach to Health Care Management, In Proceedings of the International Journal of Applied Artificial Intelligence, pp. 401-420, (1995).

[Indrusiak, 03] L. S. Indrusiak, et al, Ubiquitous Access to Reconfigurable Hardware: Application Scenarios and Implementation Issues, In Proceedings of the Design, Automation and Test in Europe Conference, (2003).

[Jacobson, 00] N. G. Jacobson, Leveraging PLDs for Embedded System Functionality, In Proceedings of the Electronics Engineer Journal, (2000).

[Jacobson, 01] N. G. Jacobson, Using the Internet to Repair Hardware in the Field, In the EETimes Magazine, July Edition, (2001).

[JADE, 05] JADE Application framework available for download at http://jade.tilab.com/

[JADE-LEAP, 05] JADE-LEAP Application framework available for download at http://jade.tilab.com/

[Jahnke, 04] J. H. Jahnke, Y. Bychkov, D. Dahlem and L. Kawasme, Implicit, Context-Aware Computing for Health Care, In

Proceedings of the Workshop on Building Software for Pervasive Computing, (2004).

[James-Roxby, 00] P. James-Roxby, S. A. Guccione, Automated Extraction of Run-Time Parameterisable Cores from Programmable Device Configurations, In IEEE Workshop on Field Programmable Custom Computing Machines, pp. 153-161, (2000).

[Jasper, 99] R. Jasper and M. Uschold, A Framework for Understanding and Classifying Ontology Applications, In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence Conference, (1999).

[Java, 05] Sun Microsystems Java White Paper available at:
http://java.sun.com/

[Java, 05b] Java Remote Method Invocation White Paper
(Sun Microsystems) available at:
http://java.sun.com/products/jdk/rmi/

[Java, 05c] Java Security API, (Sun Microsystems), available at: http://java.sun.com/security/

[Jbits, 05] Xilinx JBits Standard Development Kit for Xilinx Virtex Devices, http://www.xilinx.com/products/jbits/

[Jeode, 05] Jeode Java Virtual Machine available at:
http://www.esmertec.com/solutions/

[Jennings, 98] N. R. Jennings, K. Sycara and M. J. Wooldridge, A Roadmap of Agent Research and Development, In Proceedings of Autonomous Agents and Multi-Agent Systems Conference, (1998).

[Jennings, 01] N. R. Jennings, An Agent-Based Approach for Building Complex Software Systems. In Communications of the ACM, vol. 44, no. 4, (2001)

[Kirn, 02] S. Kirn, Ubiquitous Healthcare: The OnkoNet Mobile Agents Architecture, In Proceedings of the International Conference NetObjectDays on Objects, Components, Architectures, Services and Applications for a Networked World, (2002).

[Koch, 03] F. L. Koch and J. Meyer, Knowledge Based Autonomous Agents for Pervasive Computing using AgentLight, In Proceedings of the ACM/IFIP/USENIX International Middleware Conference, (2003).

[Koch, 04] F. Koch and I. Rahwan, Classification of Agents-based Mobile Assistants, In Proceedings of the Autonomous Agents and Multi-Agent Systems Workshop on Agents for Ubiquitous Computing, pp. 7-38, (2004).

[Kosmatos, 04] E. Kosmatos, et al, Enamorado: An intelligent multimedia content delivery system, In Proceedings of the 13th Mobile and Wireless Communications Conference, (2004).

[Kotz, 97] D. Kotz, et al, Agent TCL: Targeting the Needs of Mobile Computers, In Proceedings of the IEEE Internet Computing Journal, vol. 1, (1997).

[Kroll, 02] M. Kroll, et al, Accessing DICOM 2D/3D-Image and Waveform Data on Mobile Devices, In Proceedings of the Second Conference on Mobile Computing in Medicine, (2002).

[Kurkovsky, 04] S. Kurkovsky, Bhagyavati, and A. Ray, A Collaborative Problem-Solving Framework for Mobile Devices, In Proceedings of the 42nd Annual ACM Southeast Conference, (2004).

[Ladas, 01] C. Ladas, R. Edwards and G. Peersman, Use of Wireless Application Protocol Service Configuration Provision over the Short Messaging System for Nomadic Device Adaptation, In Proceedings of the 2nd Symposium on the Convergence of Telecommunications, Networking and Broadcasting, (2001).

[LaMarca, 05] A. LaMarca, et al, Place Lab: Device Positioning using Radio Beacons in the Wild, In Proceedings of the Pervasive 2005 Conference, Munich, Germany, (2005).

[Lamberti, 02] F. Lamberti, et al, A Web-based Architecture Enabling Multichannel Telemedicine Applications, In Proceedings of Systemics, Cybernetics and Informatics, pp. 257-262, (2002).

[Lee, 00] S. Lee, K. Yun, K. Choi. S. Hong, S. Moon and J. Lee, Java-Based Programmable Networked Embedded System Architecture with Multiple Application Support, In Proceedings of the Chip Design Automation Conference, (2000).

[Li, 00] Y. Li, T. Callahan, E. Darnell, R. Harr, U. Kurkure, and J. Stockwood, Hardware-Software Co-design of Embedded Reconfigurable Architectures, In Proceedings of the Design Automation Conference, (2000).

[Lino, 03] N. Q. Lino, et al, Delivering Intelligent Planning Information to Mobile Device Users in Collaborative

Environments, In Proceedings 18th International Joint Conference on Artificial Intelligence, AI Moves to IA: Workshop on Artificial Intelligence, Information Access, and Mobile Computing, (2003).

[Luck, 03] M. Luck, P. McBurney and C. Preist, Agent Technology: Enabling Next Generation Computing, A Roadmap for Agent-Based Computing, available at http://www.agentlink.org, (2003).

[Ma, 03] J. Ma, Incremental Design Techniques with Non-Preemptive Refinement for Million-Gate FPGAs, PhD Dissertation, Department of Electrical and Computer Engineering, Virginia Institute of Technology, (2003).

[Mahmoud, 01] Q. H. Mahmoud, MobiAgent: An Agent-based Approach to Wireless Information Systems, In Proceedings of the Third International Conference Bi-Conference Workshop on Agent-Oriented Information Systems, (2001).

[Mahmoud, 02] Q. H. Mahmoud and L. Vasiu, Accessing and Using Internet Services from Java-Enabled Handheld Wireless Devices: A Mediator-based Approach, In Proceedings of the 4th International Conference on Enterprise Information Systems, pp. 1048-1053, (2002).

[Mano, 02] M. M. Mano, Digital Design, Third Edition, Prentice-Hall Publishers, London, (2002).

[Mapen, 99] B. E. Mapen, et al, Integrating Adaptive Computing with Distributed Services Medicine, In Proceedings of the 12th IEEE Symposium on Computer-Based Medical Systems, (1999).

[Marques, 01] P. Marques, et al, Providing Applications with Mobile Agent Technology, In Proceedings of the Fourth IEEE International Conference on Open Architectures and Network Programming Conference, (2001).

[Martel, 00] S. Martel, K. Doyle and I. Hunter, Reducing the Cost, Size, and Power Consumption of Medical Devices through Communication and Reconfigurable Computing Techniques, In Proceedings of the IEEE-EBMS Asia Pacific Conference on Biomedical Engineering, China, (2000).

[Mascolo, 02] C. Mascolo, L. Capra and W. Emmerich, Middleware for Mobile Computing, In Advanced Lectures on Networking, Editors: E. Gregori, G. Anastasi, and S. Basagni, vol. 2497, pp. 20-58, (2002).

[Mascolo, 04] C. Mascolo, L. Capra and W. Emmerich, Principles of Mobile Computing Middleware, In Middleware for Communications, Editor: Q. Mahmoud, John Wiley, (2004).

[Mei, 00] B. Mei, P. Schaumont, and S. Vernalde, A Hardware-Software Partitioning and Scheduling Algorithm for Dynamically Reconfigurable Embedded Systems, In Proceedings of the Circuits, Systems and Signal Processing Workshop, (2000).

[Mena, 00] E. Mena, A. Illarramendi and A. Goni, A Software Retrieval Service based on Knowledge-Driven Agents, In Proceedings of the 7th International Conference on Cooperative Information Systems, pp. 174-185, (2000).

[Mena, 02] E. Mena, J. A. Royo, A. Illarramendi and A. Goni, An Agent-based Approach for Helping Users of Hand-Held Device to Browse Catalogs, In Proceedings of the 6th International Cooperative Information Agents Workshop, pp. 51-65, (2002).

[Metropolis, 49] N. Metropolis, and U. Stanislaw, The Monte Carlo method, Journal of the American Statistical Association, 44 (247), pp. 335-341, (1949).

[Mignolet, 02] J. Y. Mignolet, S. Vernalde, D. Verkest, and R. Lauwereins, Enabling Hardware-Software Multitasking on a Reconfigurable Computing Platform for Networked Portable Multimedia Appliances, In Proceedings of the International Conference on Engineering Reconfigurable Systems and Architectures, (2002).

[Mignolet, 03] J. Y. Mignolet, et al, Infrastructure for Design and Management of Relocatable Tasks in a Heterogeneous Reconfigurable System-on-Chip, In Proceedings of the Design, Automation and Test in Europe (DATE) Conference, pp. 986-991, (2003).

[Misra, 99] P. Misra, B. P. Burke, and M. M. Pratt, GPS Performance in Navigation, In Proceedings of the IEEE (Special Issue on GPS), vol. 87, pp. 65-85, (1999).

[Moreno, 03] A. Moreno, A. Valls and A. Viejo, Using JADE-LEAP to Implement Agents in Mobile Devices, Research Report 03-008, Technical School of Engineering, Universitat Rovira i Virgili, Spain, (2003).

[Moseley, 02] R. Moseley, Reconnetics: A System for the Dynamic Implementation of Mobile Hardware Processes in FPGAs, In Proceedings of the Communicating Process Architectures Journal, (2002).

[Moulin, 96] B. Moulin and M. Brassad, A scenario-based design method and an environment for the development of multiagent systems, In Proceedings of the First Australian Workshop on Distributed Artificial Intelligence, (1996).

[Neff, 03] N. Neff and G. Sampath, An Object Framework for Teaching ALU Component Design in Architecture Courses, In Proceedings of the Journal of Computing Sciences in Colleges, vol. 18, no. 5, pp. 23-30, (2003).

[Nikolouzou, 04] E. Nikolouzou, et al, Real-time multimedia content delivery system for nomadic users, In Proceedings of the 1st International Workshop on Streaming Media Distribution over the Internet, (2004).

[Nikolouzou, 04b] E. Nikolouzou, et al, Constraint-based Media Content Deivery over Heterogeneous Networks and Devices**,** In Proceedings of the International Journal of Wireless and Mobile Computing, Special Issue on Media streaming over Wireless and Mobile networks, (2004).

[Nitsch, 03] C. Nitsch, C. Lara and U. Kebschull, A Novel Design Technology for Next Generation Ubiquitous Computing Architectures, In Proceedings of the Reconfigurable Architectures Workshop, (2003)

[Nwana, 98] H. S. Nwana, D. T. Ndumu and L. C. Lee, Zeus: An Advanced Toolkit for Engineering Distributed Multi-Agent Systems, In Proceedings of the International Conference on the Practical Application of Intelligent Agents and Multi-Agent Systems, pp. 377-391, (1998).

[Odell, 01] J. Odell, H. Van Dyke Parunak and B. Bauer, Representing Agent Interaction Protocols in UML, Agent-Oriented Software Engineering, Editors: P. Ciancarini and M. Wooldridge, Springer-Verlag, Berlin, pp. 121-140, (2001).

[O' Sullivan, 06a] T. O' Sullivan and R. Studdert, Learning Agents on Handheld Devices Negotiating for Reconfigurable Resources, (2006). [In Preparation]

[O' Sullivan, 06b] T. O' Sullivan and R. Studdert, Context-Aware Hardware Reconfiguration of Handheld Devices, (2006). [In Preparation]

[O' Sullivan, 06c] T. O' Sullivan and R. Studdert, Collaborative Learning Amongst Handheld Devices Benefiting Negotiation for Networked Reconfigurable Resources, (2006). [In Preparation]

[O' Sullivan, 06d] T. O' Sullivan, J. O' Donoghue, J. Herbert, and R. Studdert, CAMMD: Context Aware Mobile Medical Devices, In Proceedings of the International Journal of Universal Computer Science, Special Issue on Pervasive Health Management: New Challenges for Health Informatics, (2006).

[O' Sullivan, 06e] T. O' Sullivan and R. Studdert, Mixed Agent-Object Design Technique, (2006). [In Preparation]

[O' Sullivan, 05a] T. O' Sullivan and R. Studdert, Context-Aware Negotiation for Reconfigurable Resources with Handheld Devices, In Proceedings of the OnTheMove Federated Conferences, Workshop on Context-Aware Mobile Systems, Ayia Napa, Cyprus, (2005).

[O' Sullivan, 05b] T. O' Sullivan, Context Aware Mobile Devices and Reconfigurable Computing, In Proceedings of the Fifth International Conference on Modelling and Using Context, Doctoral Colloquium, Paris, France, (2005).

[O' Sullivan, 05c] T. O' Sullivan and R. Studdert, Handheld Medical Devices Negotiating for Reconfigurable Resources using Agents, In Proceedings of the 18th IEEE International Symposium on Computer-Based Medical Systems, Dublin, Ireland, (2005).

[O' Sullivan, 05d] T. O' Sullivan and R. Studdert, Agent Technology and Reconfigurable Computing for Mobile Devices, In 20th ACM Symposium on Applied Computing, Special Track on Handheld Computing, Santa Fe, New Mexico, USA, (2005).

[O' Sullivan, 05e] T. O' Sullivan and R. Studdert, MMD-ARC: Mobile Medical Devices Incorporating Agents and Reconfigurable Computing, In Proceedings of the IEEE Conference on Enabling Technologies for Smart Appliances, Hyderabad, India, (2005).

[O' Sullivan, 04a] T. O' Sullivan and R. Studdert, Configuration Management for Networked Reconfigurable Embedded Devices, In Proceedings of the IEEE/IFIP Mobility Aware Technologies & Applications, Florianopolis, Brazil, (2004).

[O' Sullivan, 04b] T. O' Sullivan, Enabling Next Generation Embedded Devices with Reconfigurable Computing and Agent Technology, In Proceedings of the Student Session of the 6th European Agent Systems Summer School, Liverpool, United Kingdom, (2004).

[O' Sullivan, 04c] T. O' Sullivan and R. Studdert, Mobile Agent Technology and Networked Reconfigurable Embedded Devices, In Proceedings of the International Conference on Pervasive Computing and Communications, Las Vegas, USA, (2004).

[Pave, 05] Pave Framework White Paper available at: http://direct.xilinx.com/bvdocs/publications/ds084.pdf

[Pering, 05] T. Pering, V. Raghunathan and R. Want, Exploiting Radio Hierarchies for Power-Efficient Wireless Device Discovery and Connection Setup, In Proceedings of the International Conference on VLSI Design, pp. 774-779, (2005).

[Place Lab, 05] Place Lab: A Privacy-Observant Location System, Framework available for download at: http://www.placelab.org/

[Pnueli, 86] A. Pnueli, Specification and Development of Reactive Systems, In Proceedings of the Information Processing 86 Journal, pp. 845-858, (1986).

[Poggi, 01] A. Poggi, G. Rimassa and M. Tomsiuolo, Multi-User and Security Support for Multi-Agent Systems, In Proceedings of the Workshop from Objects to Agents, pp. 8-13, (2001).

[Pokahr, 03] A. Pokahr, L. Braubach and W. Lamersdorf, Jadex: Implementing a BDI-Infrastructure for JADE Agents, In Proceedings of the EXP Journal, vol. 3, no. 3, pp. 76-85, (2003).

[Prophet, 04] G. Prophet, Reconfigurable Systems: Shape Up for Diverse Application Tasks, In Proceedings of EDN Europe Magazine, Editor: Graham Prophet, January Edition, (2004) [Quinn, 03] H. Quinn, L. A. Smith King, M. Leeser, and W. Meleis, Runtime Assignment of Reconfigurable Hardware Components for Image Processing Pipelines, In Proceedings of the 11th Annual IEEE Symposium on Field Programmable Custom Computing Machines, (2003).

[Rachakonda, 95] R. V. Rachakonda, High-Speed Region Detection and Labelling using an FPGA-based Custom Computing Platform, In Lecture Notes in Computer Science 975 – Field-Programmable Logic and Applications, Eds. W. Moore, W. Luk, pp. 86-93, (1995).

[Raibulet, 00] C. Raibulet and C. Demartini, Mobile Agent Technology for the Management of Distributed Systems – A Case Study, In Proceedings of the TERENA Networking Conference, (2000).

[Ramparano, 02] F. Ramparano and O. Boissier, Smart Devices Embedding Multi-Agent Technologies for a Pro-Active World, In Proceedings of the Ubiquitous Computing Workshop, (2002).

[Reinhartz-Berger, 02] I. Reinhartz-Berger, D. Dori and S. Katz, Modeling Code Mobility Paradigms in OPM/Web, In Proceedings of the Israeli Workshop on Programming Languages & Development Environment, (2002).

[Rimassa, 03] G. Rimassa, Runtime Support for Distributed Multi-Agent Systems, PhD Thesis, University of Parma, Italy, (2003).

[Rodriguez, 04a] M. Rodriquez, et al, Location-Aware Access to Hospital Information and Services, In Proceedings of the IEEE

Transactions on Information Technology in Biomedicine, Vol. 8, No. 4, pp. 448-455, (2004).

[Rodriguez, 04b] M. Rodriquez, et al, An Agent Middleware for Supporting Ambient Intelligence for Healthcare, In Proceedings of the ECAI Workshop on Agents Applied in Health Care, Valencia, Spain, (2004).

[Rubinstein, 81] R. Y. Rubinstein, Simulation and the Monte Carlo method, John Wiley & Sons, New York, (1981).

[Sahai, 98] A. Sahai and C. Morin, Mobile Agents for Enabling Mobile User Aware Applications, In Proceedings of the Autonomous Agents Conference, (1998).

[Satyanarayanan, 96] M. Satyanarayanan, Fundamental Challenges in Mobile Computing, In Proceedings of the 15th ACM Symposium on Principles of Distributed Computing, (1996).

[Scalera, 00] J. Scalera and M. Jones, A Run-Time Reconfigurable Plug-In for the Winamp MP3 Player, In Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, (2000).

[Schilit, 03] B. Schilit, et al, Challenge: Ubiquitous Location-Aware Computing and the Place Lab Initiative, In Proceedings of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN, (2003).

[Schmit, 95] H. Schmit, and D. Thomas, Implementing Hidden Markov Modelling and Fuzzy Controllers in FPGAs, In Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines, (1995).

[Schmidt, 02] A. Schmidt, Ubiquitous Computing - Computing in Context, PhD Thesis, Computing Department, University of Lancaster, United Kingdom, (2002).

[Sheridan, 98] T. B. Sheridan, Rumination on Automation, In Proceedings of the 7th Symposium on Analysis, Design and Evaluation of Man-Machine Systems, (1998).

[Silva, 98] A. Silva and J. Delgado, The Agent Pattern for Mobile Agent Systems, In Proceedings of the Third European Conference on Pattern Languages of Programming and Computing, (1998).

[Silva, 99] M. L. Silva and L. Almeida, The Advantages of Using Mobile Agents in Software for Telecommunications, In Proceedings of the International Conference on Computer Communication, (1999).

[Singh, 94] S. Singh and P. Bellec, Virtual Hardware for Graphics Applications using FPGAs, In Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines, (1994).

[Smit, 99] G. J. M. Smit, et al, Reconfigurable Mobile Multimedia Systems, In Proceedings of the ProRISC workshop on Circuits, Systems and Signal Processing, pp. 431-436, (1999).

[Smit, 01] G. J. M. Smit, et al, Future Mobile Terminals: Efficiency by Adaptivity, In Proceedings of the Workshop on Mobile Communications in Perspective, (2001).

[Smit, 02] G. J. M. Smit, et al, Dynamic Reconfiguration in Mobile Systems, In Proceedings of the 12th International Conference on Field Programmable Logic and Application, (2002).

[Smith, 80] R. Smith, “The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver”, In IEEE Transactions on Computers, vol. 29, pp. 1104-1113, (1980)

[Smith King, 01] L. A. Smith King, et al, Run-Time Execution of Reconfigurable Hardware in a Java Environment, In Proceedings of the International Conference on Computer Design, pp. 380-385, (2001).

[Spiegel, 01] J. Van der Spiegel, VHDL Tutorial, Department of Electrical Engineering, University of Pennsylvania, USA, (2001) available at: http://www.seas.upenn.edu/~ee201/

[Sterbenz, 02] J. P. G. Sterbenz, et al, Survivable Mobile Wireless Networks: Issues, Challenges, and Research Directions, In Proceedings of the ACM Workshop on Wireless Security, (2002).

[Styles, 00] H. Styles and W. Luk, Customising Graphics Applications: Techniques and Programming Interface, In Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines, (2000).

[Sun, 05a] Sun Microsystems, Java Authentication and Authorization Service (JAAS) Reference Guide, available at: http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html

[Sun, 05b] Sun Microsystems, Java Cryptography Extension (JCE) Reference Guide, available at: http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html

[Sun, 05c] Sun Microsystems, Java Secure Socket Extension (JSSE) Reference Guide, available at: http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html

[Sweeney, 02] C. Sweeney, Hardware Design Methodologies, Celoxica University Press, (2002).

[Tarkoma, 02] S. Tarkoma and M. Laukkanen, Facilitating Agent Messaging on PDAs, In Fourth International Workshop on Mobile Agents for Telecommunication Applications, (2002).

[Taylor, 99] R. Reed Taylor and S. Copen Goldstein, A High-Performance Flexible Architecture for Cryptography, In Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems, pp. 231-245, (1999).

[Todman, 05] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk and P. Y. K. Cheung, Reconfigurable Computing: Architectures and Design Methods, In Proceedings of the IEE Computer and Digital Techniques Journal, vol. 152, no. 2, pp. 193-208, (2005).

[Tolksdorf, 98] R. Tolksdorf, Coordination Patterns of Mobile Information Agents, In Proceedings of the Cooperative Information Agents Workshop, (1998).

[Uschold, 98] M. Uschold, Knowledge Level Modelling: Concepts and Terminology, In Proceedings of the Knowledge Engineering Review, vol. 13, no. 1, (1998).

[Vasilko, 01] M. Vasilko, L. Machacek, M. Matej, P. Stepien and S. Holloway, A Rapid Prototyping Methodology and Platform for Seamless Communication Systems, In Proceedings of the 12th IEEE International Workshop on Rapid System Prototyping, pp. 70-76, (2001).

[Verbauwhede, 02] I. Verbauwhede and M. Chang, Reconfigurable Interconnect for Next Generation Systems, In Proceedings of the International Workshop on System-Level Interconnect Prediction, (2002).

[Vitaglione, 02] G. Vitaglione, F. Quarta and E. Cortese, Scalability and Performance of JADE Message Transport System, In Proceedings of Autonomous Agents and Multi-Agent Systems Conference, (2002).

[Vitaglione, 02b] Jade Security Administrator Guide, University of Parma, September 2002 (v 2.61), available at: http://sharon.cselt.it/projects/jade/

[Vuillemin, 96] J. Vuillemin, et al, Programmable Active Memories: Reconfigurable Systems Come of Age, In Proceedings of IEEE Transactions on VLSI Systems, vol. 4, no. 1, pp. 56-69, (1996).

[Wang, 04] G. Wang, et al, Application of Middleware Technologies to Mobile Enterprise Information Service, In Middleware for Communications: Concepts, Designs, & Case Studies, Editor: Q. Mahmoud, Wiley Publications, (2004).

[Weiser, 91] M. Weiser, The Computer for the 21st Century, In Proceedings of the Scientific American, vol. 265, no. 3, pp. 94-104, (1991).

[Weiser, 93] M. Weiser, Some Computer Science Issues in Ubiquitous Computing, In Proceedings of the Communications of the ACM, vol. 36, no. 7, (1993).

[Weiss, 02] G. Weiss, Agent Orientation in Software Engineering, In Proceedings of the Knowledge Engineering Review, vol. 16, no. 4, pp. 349-373, (2002).

[Weiss, 04] M. Weiss, A Pattern Language for Motivating the Use of Agents, In Proceedings of the Agent-Oriented Information Systems Conference, (2004).

[Wind, 05] Wind River Systems: http://www.windriver.com/

[Wirthlin, 02] M. J. Wirthlin and B. McMurtrey, IP Delivery for FPGAs using Applets and JHDL, In Proceedings of the Design Automation Conference, (2002).

[Wood, 00] M. Wood and S. A. DeLoach, An overview of the multi-agent systems engineering methodology, In Proceedings of the First International Workshop on Agent-Oriented Software Engineering, (2000).

[Wooldridge, 95] M. Wooldridge and N. R. Jennings, Intelligent Agents: Theory and Practise, In Proceedings of the Knowledge Engineering Review, vol. 10, no. 2, pp. 115-152, (1995).

[Wooldridge, 97] M. Wooldridge, Agent-Based Software Engineering, In Proceedings of IEE Journal of Software Engineering, vol. 144, pp. 26-37, (1997).

[Wooldridge, 98] M. Wooldridge and N. R. Jennings, Pitfalls of Agent-Oriented Development, In Proceedings of the Second International Conference on Autonomous Agents, pp. 385-391, (1998).

[Wooldridge, 99] M. Wooldridge, N. R. Jennings, and D. Kinny, A Methodology for Agent-Oriented Analysis and Design, In Proceedings of the Third International Conference on Autonomous Agents, (1999).

[Wooldridge, 02] M. Wooldridge, An Introduction to MultiAgent Systems, John Wiley & Sons, West Sussex, England, (2002).

[Xilinx, 01] Architecting Systems for Upgradeability with IRL (Internet Reconfigurable Logic) 2001. Xilinx Application Note, XAPP 412 (v1.0).

[Xilinx, 02] The Programmable Logic Data Book, Xilinx Corporation Publications, (2002).

[Xilinx, 03] Xilinx Virtex Series Configuration Architecture, User Guide, (2003).

[Xilinx, 05] Xilinx Corporation: http://www.xilinx.com

[Xilinx, 05b] Xilinx Integrated Software Development Environment, version 7.1, available at: http://www.xilinx.com

[Xu, 00] C. Xu and B. Wims, A Mobile Agent Based Push Methodology for Global Parallel Computing, In

Proceedings of the Wiley Journal: Concurrency – Practice and Experience, vol. 12, pp. 705-726, (2000).

[Yokota, 02] T. Yokota, et al, A Scalable FPGA-based Custom Computing Machine for Medical Image Processing, In Proceedings of the 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, (2002).

[Zaslavsky, 04] A. Zaslavsky, Mobile Agents: Can They Assist with Context Awareness, In Proceedings of the IEEE International Conference on Mobile Data Management, (2004).

APPENDIX

Appendix A.1

```
// Gray Scale Image Filter implemented with JBits

import  java.io.*;

import com.xilinx.JBits.CoreTemplate.Bitstream;
import com.xilinx.JBits.CoreTemplate.CoreOutput;
import com.xilinx.JBits.CoreTemplate.Offset;
import com.xilinx.JBits.CoreTemplate.Gran;
import com.xilinx.JBits.CoreTemplate.Net;
import com.xilinx.JBits.CoreTemplate.Bus;

import com.xilinx.JBits.Virtex.Devices;
import com.xilinx.JBits.Virtex.JBits;

import com.xilinx.JRoute2.Virtex.JRoute;

import com.xilinx.JBits.Virtex.RTPCore.Arithmetic.Subtracter;
import com.xilinx.JBits.Virtex.RTPCore.Basic.Constant;
import com.xilinx.JBits.Virtex.RTPCore.Basic.Counter;
import com.xilinx.JBits.Virtex.RTPCore.Basic.CounterProperties;
import com.xilinx.JBits.Virtex.RTPCore.Basic.MemoryElements.BRAM;
import
com.xilinx.JBits.Virtex.RTPCore.Basic.MemoryElements.BRAMPropertie
s;
import com.xilinx.JBits.Virtex.RTPCore.Basic.Clock;
import com.xilinx.JBits.Virtex.RTPCore.Basic.Register;

import com.xilinx.JBits.Virtex.Devices;
import com.xilinx.JBits.CoreTemplate.*;
import com.xilinx.Netlist.SYM.*;

/**
**  This application tests a GrayScale filter core.
**  8-bit red, green and blue inputs for the filter come from
**  three BRAM.  The filter output goes to another BRAM in the
**  same column.  A simple counter core feeds the address lines of
**  the input BRAM and a delayed version of the address
**  (accounting for BRAM latency and processor pipeline delay)
**  feeds the output BRAM address lines.
**  The inputs and output of the filter are registered.
*/


public class GrayScaleProcessorTest extends RTPCore
{
public GrayScaleProcessorTest(String name) throws CoreException
{
        super(name);

        int height = Devices.getClbRows(Devices.XCV1000);
        int width = Devices.getClbColumns(Devices.XCV1000);
        setHeight(calcHeight(height));
        setWidth(calcWidth(width));
        setHeightGran(calcHeightGran());
        setWidthGran(calcWidthGran());
} // end of constructor

// return the vertical granularity of this core
public static int calcHeightGran()
{
        return Gran.CLB;
} // end of calcHeightGran() method
```

```
// return the horizontal granularity of this core.
public static int calcWidthGran()
{
        return Gran.CLB;
} // end of calcWidthGran() method

// calculates height for this core.
public static int calcHeight(int height)
{
        //return 16;
        return height;
} // end of calcHeight() method

// calculates width for this core.
public static int calcWidth(int width)
{
        //return 20;
        return width;
} // end of calcWidth() method

/** This method creates and places the RTPCores */
public void implement()
{
        // Define the location parameters
        int row = 0;
        int col = 2;
        int bramRow = 0;
        int bramCol = 0;

        // Define the data path widths
        int addressGeneratorWidth = 10;
        // Only need nine, but has to be even number
        int addressWidth = 9;
        // because Counter core only takes even number
        int dataWidth = 8;

        // Define the Ram size in total bits
        int ramSize = 4096/dataWidth;

        //
        // Create top level counter nets and buses
        //
        Net clk = new Net("clk", null);
        Bus addressGeneratorOut = new Bus("addressGeneratorOut",
null, addressGeneratorWidth);

        Bus address = new Bus("address", null, addressWidth);

        Bus delayAddress = new Bus("Delayed address", null,
addressWidth);

        Bus constantDelay = new Bus("Constant value", null,
addressWidth);

        Bus red = new Bus("Red Input reg",null,dataWidth);
        Bus blue = new Bus("Bl Input reg",null,dataWidth);
        Bus green = new Bus("Gr Input reg",null,dataWidth);

        Bus gray = new Bus("Gray Input reg",null,dataWidth);

        Bus grayOut = new Bus("Op reg",null,dataWidth);

        Bus dummyAddressInRamB1 = new Bus("dummyAddressInRamB",
null, addressWidth);

        Bus dummyInputRamInA1 = new Bus("dummyInputRamInA", null,
dataWidth);
        Bus dummyInputRamInB1 = new Bus("dummyInputRamInB", null,
dataWidth);
```

```
        Bus dummyInputRamOutB1 = new Bus("dummyInputRamOutB",
null, dataWidth);

        Bus dummyAddressOutRamB = new Bus("dummyAddressOutRamB",
null, addressWidth);
        Bus dummyDIBOutRam = new Bus("dummyDIBOutRam", null,
dataWidth);
        Bus dummyOutputRamOutA = new Bus("dummyOutputRamOutA",
null, dataWidth);
        Bus dummyOutputRamOutB = new Bus("dummyOutputRamOutB",
null, dataWidth);

// Define the output buses for the read-only RAM.
// These buses are routed to other inputs in the circuit.
Bus inputRamOutA1 = new Bus("inputRamOutA1", null, dataWidth);
Bus inputRamOutA2 = new Bus("inputRamOutA2", null, dataWidth);
Bus inputRamOutA3 = new Bus("inputRamOutA3", null, dataWidth);

try
{
// Connect the addressGeneratorOutput (n-1 outputs) to the address
// bus
// Don't need the last odd connection.
for (int i = 0; i < addressWidth; i++)
{
        addressGeneratorOut.setNet(i, address.getNet(i));
} // end of for

addressGeneratorOut.setNet(addressGeneratorWidth-1,
Net.NoConnect);

//
//  Create the Cores
//

// Create Clock Core
Clock clock = new Clock("clock", clk);

// Create Counter Properties for the address generator
CounterProperties cpAddressGenerator = new CounterProperties();
cpAddressGenerator.setIn_clk(clk);
cpAddressGenerator.setIn_ce(Net.NoConnect);

cpAddressGenerator.setIn_rst(Net.NoConnect);
cpAddressGenerator.setOut_dout(addressGeneratorOut);

// Create the address generator
Counter addressGenerator = new Counter("addressGenerator",
cpAddressGenerator);

// Create an address adjusting circuit
Subtracter delayGenerator = new Subtracter("Delay", address,
constantDelay, delayAddress);

// Create a constant delay
Constant delay = new Constant("Delay Constant", constantDelay);

// Create properties for the input RAM
BRAMProperties inputRamProperties1 = new BRAMProperties();
inputRamProperties1.setADDRA(address);
inputRamProperties1.setADDRB(dummyAddressInRamB1);
inputRamProperties1.setCLKA(clk,false);
inputRamProperties1.setDIA(dummyInputRamInA1);
inputRamProperties1.setDIB(dummyInputRamInB1);
inputRamProperties1.setDOA(inputRamOutA1);

inputRamProperties1.setDOB(dummyInputRamOutB1);
inputRamProperties1.setWEA(0);
inputRamProperties1.setWEB(0);
inputRamProperties1.setSELA(1);
```

```
inputRamProperties1.setSELB(0);
inputRamProperties1.setRSTA(0);
inputRamProperties1.setRSTB(0);

// Create properties for the input RAM
BRAMProperties inputRamProperties2 = new BRAMProperties();
inputRamProperties2.setADDRA(address);
inputRamProperties2.setADDRB(dummyAddressInRamB1);
inputRamProperties2.setCLKA(clk,false);
inputRamProperties2.setDIA(dummyInputRamInA1);
inputRamProperties2.setDIB(dummyInputRamInB1);
inputRamProperties2.setDOA(inputRamOutA2);

inputRamProperties2.setDOB(dummyInputRamOutB1);
inputRamProperties2.setWEA(0);
inputRamProperties2.setWEB(0);
inputRamProperties2.setSELA(1);
inputRamProperties2.setSELB(0);
inputRamProperties2.setRSTA(0);
inputRamProperties2.setRSTB(0);

// Create properties for the input RAM
BRAMProperties inputRamProperties3 = new BRAMProperties();
inputRamProperties3.setADDRA(delayAddress);
inputRamProperties3.setADDRB(dummyAddressInRamB1);
inputRamProperties3.setCLKA(clk,false);
inputRamProperties3.setDIA(dummyInputRamInA1);
inputRamProperties3.setDIB(dummyInputRamInB1);
inputRamProperties3.setDOA(inputRamOutA3);

inputRamProperties3.setDOB(dummyInputRamOutB1);
inputRamProperties3.setWEA(0);
inputRamProperties3.setWEB(0);
inputRamProperties3.setSELA(1);
inputRamProperties3.setSELB(0);
inputRamProperties3.setRSTA(0);
inputRamProperties3.setRSTB(0);


// Create the Input RAM
BRAM inputRam1 = new BRAM("red input RAM", inputRamProperties1);
BRAM inputRam2 = new BRAM("green input RAM", inputRamProperties2);
BRAM inputRam3 = new BRAM("blue input RAM", inputRamProperties3);

// Create properties for the output RAM
BRAMProperties outputRamProperties = new BRAMProperties();
outputRamProperties.setADDRA(delayAddress);
outputRamProperties.setADDRB(dummyAddressOutRamB);
outputRamProperties.setCLKA(clk,false);
outputRamProperties.setDIA(grayOut);
outputRamProperties.setDIB(dummyDIBOutRam);

outputRamProperties.setDOA(dummyOutputRamOutA);
outputRamProperties.setDOB(dummyOutputRamOutB);
outputRamProperties.setWEA(1);
outputRamProperties.setWEB(0);
outputRamProperties.setSELA(1);
outputRamProperties.setSELB(0);
outputRamProperties.setRSTA(0);
outputRamProperties.setRSTB(0);

// Create the Output RAM
BRAM outputRam = new BRAM("output Ram", outputRamProperties);
// Create Register Cores
Register reRegister = new Register("Red Input
Register",clk,inputRamOutA1,red);
Register blRegister = new Register("Blue Input
Register",clk,inputRamOutA2,blue);
Register grRegister = new Register("Green Input
Register",clk,inputRamOutA3,green);
```

```
Register opRegister = new Register("Op Input
Register",clk,gray,grayOut);

// Create GrayScale Core
GrayScale filter = new
GrayScale("GrayScale1",clk,red,green,blue,gray);

//
// Set Offset Locations so we can place the cores
//
// Set the input counter offset

reRegister.getRelativeOffset().setVerOffset(Gran.CLB, row);
reRegister.getRelativeOffset().setHorOffset(Gran.CLB, col);
blRegister.getRelativeOffset().setVerOffset(Gran.CLB, row+4);
blRegister.getRelativeOffset().setHorOffset(Gran.CLB, col);
grRegister.getRelativeOffset().setVerOffset(Gran.CLB, row+8);
grRegister.getRelativeOffset().setHorOffset(Gran.CLB, col);
opRegister.getRelativeOffset().setVerOffset(Gran.CLB, row+12);

opRegister.getRelativeOffset().setHorOffset(Gran.CLB, col);
// Set the addressGenerator offset
addressGenerator.getRelativeOffset().setVerOffset(Gran.CLB, row);
addressGenerator.getRelativeOffset().setHorOffset(Gran.CLB,
col+1);

// Set the delayGenerator offset
delayGenerator.getRelativeOffset().setVerOffset(Gran.CLB, row+12);
delayGenerator.getRelativeOffset().setHorOffset(Gran.CLB, col+1);

// Set the delay offset
delay.getRelativeOffset().setVerOffset(Gran.CLB, row+6);
delay.getRelativeOffset().setHorOffset(Gran.CLB, col+1);
delay.getRelativeOffset().setVerOffset(Gran.CLB, row+6);
delay.getRelativeOffset().setHorOffset(Gran.CLB, col+1);

// Set the filter offset
filter.getRelativeOffset().setVerOffset(Gran.CLB, row);
filter.getRelativeOffset().setHorOffset(Gran.CLB, col+2);

// Create the initial data for the input Ram
int[] initialInputRamValues1 = new int[ramSize];
int[] initialInputRamValues2 = new int[ramSize];
int[] initialOutputRamValues = new int[ramSize];

for (int i = 0; i < ramSize; i++)
{
        initialInputRamValues2[i] = 0;
        initialOutputRamValues[i] = 0;
} // end of forss

// Implement the RTPCores
clock.implement(1); // Use GCLK3 for XCV1000, GCLK1 otherwise
addressGenerator.implement();
delayGenerator.implement();
delay.implement((long) 13);  // pipeline and BRAM delay = 13
reRegister.implement();
blRegister.implement();
grRegister.implement();
opRegister.implement();
filter.implement();
inputRam1.implement(bramRow, bramCol, initialInputRamValues2);
inputRam2.implement(bramRow+1, bramCol, initialInputRamValues2);
inputRam3.implement(bramRow+2, bramCol, initialInputRamValues2);
outputRam.implement(bramRow+3, bramCol, initialOutputRamValues);

//
// Connect the buses/nets
//
Bitstream.connect(clk);
Bitstream.connect(inputRamOutA3 Bitstream.connect(inputRamOutA1);
Bitstream.connect(inputRamOutA2);

Bitstream.connect(red);
Bitstream.connect(blue);
Bitstream.connect(green);
Bitstream.connect(gray);
Bitstream.connect(grayOut);
Bitstream.connect(address);
Bitstream.connect(delayAddress);
Bitstream.connect(constantDelay);
} // end of class
```

Appendix A.2

```
// Handel-C Edge Detection Algorithm

// platform specific definitions
// board Type
#define PP1000_BOARD_TYPE PP1000_V2_VIRTEX

// handel-c clock is the same frequency as RC1000-PP clock
#define PP1000_DIVIDE1

// PP1000 clock is PP1000_MCLK
#define PP1000_CLOCK PP1000_MCLK

// clock rate
#define PP1000_CLOCKRATE 20

// use 8 bit external RAM access - but can also use 32 bit
#define PP1000_32BIT_RAMS

// memory bank details
// source pixel memory bank data (bank 0: mask = 0b0001 = 0x1)
#define SOURCE_PIXEL_MEMORY_BANK 0b1111
// destination pixel memory bank data (bank 3: mask = 0b1000 =
0x8)
//#define DESTINATION_PIXEL_MEMORY_BANK 0x8

// paramters of image and threshold for edges
#define LOG2_WIDTH 8
#define WIDTH 256
#define LOG2_HEIGHT 8
#define HEIGHT 256
#define TOP_THRESHOLD 16
#define PIXEL_NUMBER 256*256

// family and part details
set family = XilinxVirtex;
set part = "XCV1000BG560-6";

// include rc1000-pp support header file
#include <pp1000.h>

// macro to determine absolute value
macro expr abs(a) = (a < 0 ? -a : a);


void main (void)
{

        unsigned int 32 originalPixelFromMem;
        unsigned int 32 leftPixel;
        unsigned int 32 belowPixel;
        unsigned int 32 originalPixel;
        unsigned int 32 resultPixel;

        signed 21 rowCounter;
        unsigned 21 columnCounter;

        // declare memory address holder
        unsigned int 21 memoryAddress;
        unsigned int 21 memoryAddressLeft;
        unsigned int 21 memoryAddressBelow;
        unsigned int 8 count;


        unsigned int 21 firstRowChecker;

        // status variable to coordinate action between host
        // and FPGA
        unsigned char status;

        // assign initial value to memory address holder
        memoryAddress = 0;

        // request source pixel memory bank
        PP1000RequestMemoryBank(SOURCE_PIXEL_MEMORY_BANK);

        // loop for every pixel
        for(rowCounter = HEIGHT; rowCounter > -1; rowCounter--)
        {
        for(columnCounter = WIDTH; columnCounter > 0;
columnCounter--)
        {
        if (rowCounter == 0)
        {
        memoryAddress = (unsigned) (columnCounter-1);
        } // end of if
        else
        {
        firstRowChecker = (unsigned) (rowCounter*WIDTH);
                memoryAddress = (unsigned) (columnCounter-1) +
                               firstRowChecker;
        } // end of else


        PP1000ReadBank0(originalPixelFromMem,memoryAddress);
        leftPixel = originalPixelFromMem;
        memoryAddress = memoryAddress + 1;

        PP1000ReadBank0(originalPixelFromMem,memoryAddress);
        originalPixel = originalPixelFromMem;

        if (rowCounter != HEIGHT)
        {
        memoryAddress = memoryAddress + 256;
        PP1000ReadBank0(originalPixelFromMem,memoryAddress);
        belowPixel = originalPixelFromMem;
        memoryAddress = memoryAddress - 256;
        } // end of if
        else
        {
        // just make belowPixel same as originalPixel
        belowPixel = originalPixel;
        } // end of else

        if(leftPixel > originalPixel)
        {
        resultPixel = leftPixel - originalPixel;
        } // end of if
        else
        {
        resultPixel = originalPixel - leftPixel;
        } // end of else

        if (resultPixel < TOP_THRESHOLD)
        {
        // check against pixel below
        if (belowPixel > originalPixel)
        {
        resultPixel = belowPixel - originalPixel;
        } // end of inner if
        else
        {
        resultPixel = originalPixel - belowPixel;
        } // end of inner else

        if (resultPixel < TOP_THRESHOLD)
        {
```

```
        PP1000WriteBank0(memoryAddress, 0);
        } // end of inner if
        else
        {
        PP1000WriteBank0(memoryAddress, 255);
        } // end of inner else
        } // end of if
        else
        {
        PP1000WriteBank0(memoryAddress, 255);
        } // end of else
        } // end of inner for
        } // end of for

        // release source pixel memory bank
        PP1000ReleaseMemoryBank(SOURCE_PIXEL_MEMORY_BANK);

        // assign value to status
        status = '2';
        PP1000WriteStatus(status);

} // end of main()
```

Appendix A.3

```
// Name: TestVirtexDS
// Written by: Timothy O' Sullivan
// Date: 30/08/03
// Description:  This class is responsible for making
//                XHWIF calls to FPGA board

// xilinx board connection classes
import  com.xilinx.XHWIF.XHWIF;
import  com.xilinx.JBits.Virtex.ConfigurationException;
import  com.xilinx.JBits.Virtex.ReadbackCommand;
import  com.xilinx.JBits.Virtex.Devices;
import  com.xilinx.JBits.Virtex.JBits;
import  com.xilinx.JBits.Virtex.ConfigurationException;
import  com.xilinx.JBits.Virtex.Bits.CLB;

// xilinx CoreTemplate classes
import com.xilinx.JBits.CoreTemplate.Bitstream;
import com.xilinx.JBits.CoreTemplate.CoreOutput;
import com.xilinx.JBits.CoreTemplate.Offset;
import com.xilinx.JBits.CoreTemplate.Gran;
import com.xilinx.JBits.CoreTemplate.Net;
import com.xilinx.JBits.CoreTemplate.Bus;
import com.xilinx.JBits.CoreTemplate.Port;
import com.xilinx.JBits.CoreTemplate.Pin;
import com.xilinx.JBits.CoreTemplate.Bitstream;
import com.xilinx.JBits.CoreTemplate.CoreException;
import com.xilinx.JRoute2.Virtex.JRoute;
import com.xilinx.JRoute2.Virtex.ResourceDB.CenterWires;
import com.xilinx.Netlist.SYM.*;
import com.xilinx.Netlist.XDL.*;

// import jade classes
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;

// import general java classes
import java.util.*;
import java.io.*;
import java.lang.*;

public class TestVirtexDS
{
// declare static variables
private final static int WIDTH = 16;
public  final static int DEPTH = 16;

// declare class variables
private int deviceType;
private int port;
private int result;
private String serverName;
private byte[] configuringBitStream;
private byte[] readingBackCommand;
private byte[] readingBackData;

/////////////////////////////////////////////
private XHWIF myBoard;
////////////////////////////////////////////

public void setDeviceType(int value)
{
        this.deviceType = value;
} // end of setDeviceType(..) method
```

```
public int getDeviceType()
{
        return this.deviceType;
} // end of getDeviceType() method

public void setPort(int value)
{
        this.port = value;
} // end of setPort(..) method

public int getPort()
{
        return this.port;
} // end of getPort() method

public void setResult(int value)
{
        this.result = value;
} // end of setResult(..) method

public int getResult()
{
        return this.result;
} // end of getResult() method

public void setServerName(String value)
{
        this.serverName = value;
} // end of setServerName(..) method

public String getServerName()
{
        return this.serverName;
} // end of getServerName() method

public void setConfiguringBitStream(byte[] value)
{
        this.configuringBitStream = value;
} // end of setConfiguringBitStream(..) method

public byte[] getConfiguringBitStream()
{
        return this.configuringBitStream;
} // end of getConfiguringBitStream() method

public void setReadingBackCommand(byte[] value)
{
        this.readingBackCommand = value;
} // end of setReadingBackCommand(..) method

public byte[] getReadingBackCommand()
{
        return this.readingBackCommand;
} // end of getReadingBackCommand() method

public void setReadingBackData(byte[] value)
{
        this.readingBackData = value;
} // end of setReadingBackData(..) method

public byte[] getReadingBackData()
{
        return this.readingBackData;
} // end of getReadingBackData() method

public void loadBitStream(JBits myJBits, String inputFileName)
{
        System.out.println("In method loadBitStream:\n" + "Reading
in: " + inputFileName);
try
{
        myJBits.read(inputFileName);
} // end of try
catch (Exception ex)
{
System.out.println("Exception caught in CounterTestVirtexDS");
System.out.println("Exception caught in loadBitStream method");
System.out.println("Could not read in " + inputFileName);
ex.printStackTrace();
System.out.println("\n\n");
System.exit(-1);
} // end of catch
} // end of loadBitStream(..) method

public void startUpVirtexDS(JBits myJBits) throws
ConfigurationException
{
// Connect to the VirtexDS
myBoard = XHWIF.Get(PathDescriptor.BOARD_NAME);

// check if we have found board
if (myBoard == null)
{
System.out.println("Can not find: " + PathDescriptor.BOARD_NAME);
} // end of if

setServerName(XHWIF.GetRemoteHostName(PathDescriptor.BOARD_NAME));
setPort(XHWIF.GetPort(PathDescriptor.BOARD_NAME));
setResult(myBoard.connect(getServerName(), getPort()));

// check if connection is established
if (getResult() != 0)
{
System.out.println("Could not connect to: " +
PathDescriptor.BOARD_NAME);
System.exit(-1);
} // end of if

System.out.println("Connected to: " + PathDescriptor.BOARD_NAME);
// reset the board

System.out.print("reseting board ... ");
myBoard.reset();
System.out.println("done");

// load bitstream
System.out.print("loading null bitstream to board ... ");
myJBits.clearPartial();
setConfiguringBitStream(myJBits.getPartial());
myBoard.setConfiguration(0, getConfiguringBitStream());
System.out.println("done");
} // end of startUpVirtexDS(..) method

public void reconfig(JBits myJBits) throws ConfigurationException
{
// Generate Parital bitstream
System.out.print("loading partial bitstream to board ... ");

setConfiguringBitStream(myJBits.getPartial());
myBoard.setConfiguration(0, getConfiguringBitStream());
System.out.println("done");
} // end of reconfig() method
} // end of class
```

Appendix A.4

```
public class SendProtocolExample
{
        // declare native methods
        public native int RC200BoardConfiguration();

        static
        {
                System.loadLibrary("SendProtocolExample");
        } // end of static

        public SendProtocolExample()
        {
        } // end of constructor

        public void start()
        {
                // declare local variables
                int myReturnValue;

                try
                {
                System.out.println("ReconfigurableInterface: Begin
                Board Reconfiguration");
                myReturnValue = RC200BoardConfiguration();
                } //end of try
                catch(Exception myException1)
                {
                        myException1.printStackTrace();
                } // end of catch
        } // end of start() method
} // end of class
```