

# **A Methodology for Modelling Mobile Agent-Based Systems**

**(Mobile agent Mobility Methodology - MaMM)**

**DIVINA A. MELOMEY**

A thesis submitted in partial fulfillment  
of the requirements of the  
University of East London  
for the degree of  
Master of Philosophy

Research undertaken in the  
School of Architecture, Computing and Engineering

2012

# Acknowledgements

*'I know the thoughts that I have toward you, says the Lord, thoughts of peace and not of evil, to give you a future and a hope'.  
(Jeremiah 29:11)*

I would like to thank my supervisory team for their support in my research.

I would also like to acknowledge the funding I received from The Allan and Nesta Charitable Trust(ANFCT), Grant reference LPG/sw/Ferguson for sponsoring part of my final year Ph.D tuition for 2007/8. I am also grateful to Letty Glaister for facilitating the grant process.

I would like to express my appreciation to Dr. Godfried Williams (University of Gloucestershire) for his interest, expertise, motivation and support in shaping and developing my understanding of research. My gratitude goes to my parents Andrew and Elizabeth Melomey, for their constant presence, love and support during this research process. I also like express my appreciation to the Melomey and Atchoe family in the UK, for their love, support and friendship during this period of research. Furthermore, I would also like to thank Madam Elizabeth Odumang and Reverend Ebenezer Odai Tetey for their encouragement, prayers and friendship.

Finally, to everyone that supported and encouraged, I say thank you.

## **Abstract**

Mobile agents are a particular type of agents that have all the characteristics of an agent and also demonstrate the ability to move or migrate from one node to another in a network environment. Mobile agents have received considerable attention from industry and the research community in recent times due to the fact that their special characteristic of migration help address issues such as network overload, network latency and protocol encapsulation. Due to the current focus in exploiting agent technology mainly in a research environment, there has been an influx of software engineering methodologies for developing multi-agent systems. However, little attention has been given to modelling mobile agents. For mobile agent-based systems to become more widely accepted there is a critical need for a methodology to be developed to address various issues related to modelling mobility of agent . This research study provides an overview of the current approaches, methodologies and modelling languages that can be used for developing multi-agent systems. The overview indicates extensive research on methodologies for modelling multi-agent systems and little on mobility in mobile agent-based systems. An original contribution in this research known as Mobile agent-based Mobility Methodology (MaMM) is the methodology for modelling mobility in mobile agent-based systems using underlying principles of Genetic Algorithms (GA) with emphasis on fitness functions and genetic representation. Delphi study and case studies were employed in carrying out this research.

## Table of Contents

List of Tables .....	vii
List of Figures .....	viii
Acknowledgements .....	viii
Chapter 1: Introduction .....	1
1.1 Existing Approaches in Mobile Agent-based Systems .....	4
1.2 Research Aim and Objectives .....	5
1.3 Research Process .....	6
1.4 Thesis Structure .....	7
Chapter 2: Literature Review .....	9
2.1 Introduction .....	9
2.2 Background Theory .....	9
2.3 Definition of Software Agents .....	10
2.4 Existing Agent Modelling Languages .....	13
2.4.1 Agent Unified Modelling Language .....	16
2.4.2 Agent Modelling Language .....	17
2.4.3 Specification Language for Agent Systems .....	18
2.4.4 Caste-Centric Agent Modelling Language and Environment .....	18
2.4.5 Autonomy Specification Language .....	19
2.4.6 Other Extensions to UML .....	20
2.4.7 AUML Deployment Diagram .....	21
2.4.8 AUML Activity Diagram Extensions .....	22

2.5 Mobile Agent .....	24
2.5.1 Multi-agent Systems Engineering Methodology .....	25
2.5.2 GAIA Methodology .....	28
2.5.3 TROPOS Approach .....	29
2.5.4 Prometheus Methodology .....	30
2.6 Mobile Agent .....	32
2.7 Traditional Software Development Models.....	35
2.7.1 Waterfall Model .....	36
2.7.2 Rapid Prototyping Model .....	37
2.7.3 Evolutionary and Iterative Model .....	37
2.7.4 Incremental Model.....	37
2.7.5 Spiral Model .....	38
2.8 Discussion.....	38
2.9 Summary.....	39
Chapter 3: Research Methodology .....	40
3.1 Introduction .....	40
3.2 Delphi Technique/Study .....	41
3.2.1 The Delphi Process .....	41
3.2.2 Selection of Experts .....	41
3.2.3 Data Collection Process .....	42
3.2.3.1 Round 1.....	42
3.2.3.2 Round 2.....	43
3.2.3.3 Round 3.....	45

3.2.4 Analysis of Delphi Study Data .....	46
3.2.5 Confidentiality of Delphi Study .....	48
3.3 Case Studies.....	48
3.3.1 Data Collection and Analysis Process.....	48
3.4 Simulation .....	49
3.5 Summary.....	50
Chapter 4: Mobile Agent Modelling Modelling .....	51
4.1 Introduction .....	51
4.2 The Mobile agent Mobility Methodology (MaMM).....	51
4.2 .1 Reasons for Developing the Mobile Agent Mobility Methodology ..	52
4.2.2 Mobility Concepts and Design Requirement Considerations.....	53
4.3 Concepts of Multi-agent Systems.....	54
4.3.1 Existing Concepts of Multi-agent Systems .....	54
4.3.2 Proposed Concepts for Describing Mobile Agents .....	57
4.4 Composition of Agent System .....	58
4.5 Phases of MaMM .....	60
4.5.1 Mobility Requirement Elicitation.....	60
4.5.2 Fitness Classification .....	64
4.5.2.1 Mathematical Modelling of Mobility Requirements .....	66
4.5.2.2 Binomial Coefficient Application to Requirements .....	74
4.5.2.3 Concepts underlying GA Problem Formulation .....	78
4.5.2.4 Representation of Chromosomes in the Mobility Problem .....	83

4.5.3 Code Transformation .....	89
4.5.4 Mobility Implementation .....	91
4.6 Simulation.....	91
4.7 Simulation of Fitness function Using GA Concepts .....	92
4.7.1 Objectives of Simulation .....	92
4.7.2 Overview of Simulation .....	92
4.8 Summary .....	94
Chapter 5: Simulation and Evaluation .....	96
5.1 Introduction .....	96
5.2 Mobility Fitness Functions Testing .....	96
5.2.1 Step 1: Problem Setup.....	97
5.2.2 Step 2: Function Option for Problem Setup .....	98
5.2.3 Step 3: Monitoring and Observation. ....	99
5.3 Fitness Function Translation.....	100
5.3.1 Test 1: Remote Method Invocation .....	100
5.3.2 Test 2: Mobility Synchronisation .....	101
5.3.3 Test 3: Rastrigin's Function .....	103
5.4 Simulation Results .....	104
5.4.1 Results for Mobility Method Invocation Function .....	104
5.4.2 Results for Mobility Synchronisation Function .....	105
5.4.3 Results for Rastrigin's Function.....	107
5.5 Evaluation of Simulation Results .....	108

5.6 Discussion of Simulation Test Results .....	111
5.7 Real Life Usefulness of Results .....	113
5.8 Summary.....	116
Chapter 6: Conclusions and Future Work.....	117
6.1 Summary .....	117
6.2 Delphi Study .....	118
6.3 Case Studies .....	118
6.4 Mobile agent-based Mobility Methodology .....	119
6.4.1 Simulation Testing and Evaluation of Results.....	121
6.5 Research Contributions .....	125
6.6 Future Work.....	127
References .....	129
Appendix A - Auto generated Codes .....	144
Appendix B - Delphi Study Data .....	148
Appendix C - Case Studies Interviews .....	155
Appendix D - Publish Papers/Journal Articles .....	161



## List of Tables

Table 2.1 Analysis of Existing Modelling Languages for MAS .....	16
Table 2.2 Existing Multi-agent Systems Methodologies .....	25
Table 4.1 Generic and Mobility Requirements.....	75
Table 5.1 Final Point Co-ordinates .....	109
Table 6.1 Existing Multi-agent Systems Methodologies and MaMM.....	121

## List of Figures

Figure 1.1 Research process.....	6
Figure 4.1 Agent-to-Mobile Agent Diagram. ....	59
Figure 4.2 Phases of MaMM. ....	60
Figure 4.3 Mobility Fitness Classification Model .....	65
Figure 4.4 Genetic Algorithm Flowchart .....	86
Figure 4.5 Mobility Design Layer Diagram.....	90
Figure 4.6 GA Optimization Tool .....	94
Figure 5.1 '@mobilityRMI' function simulation Setup.....	97
Figure 5.2 '@mobilitysync' Problem Setup .....	97
Figure 5.3 '@rastriginfcn' Problem Setup .....	98
Figure 5.4 Population Options .....	98
Figure 5.5 GA Tool GUI for Genetic Operator Options .....	99
Figure 5.6 '@mobilityRMI' Results.....	104

Figure 5.7 '@mobilityRMI' function plot .....	105
Figure 5.8 '@mobilitysync' function option .....	105
Figure 5.9 '@mobilitysync' function plot .....	106
Figure 5.10 '@mobilitysync' function results.....	106
Figure 5.11 '@rastriginsfcn' simulation results .....	107
Figure 5.12 '@rastriginsfcn' simulation function plot.....	108
Figure 5.13 '@mobilitysync' plot function .....	110
Figure 5.14 '@mobilityRMI' plot function .....	110
Figure 5.15 '@rastriginsfcn' plot function.....	110

# CHAPTER 1

## Introduction

With the emergence of mobile and wireless information communication technology, the focus of research in distributed computing has turned to addressing challenges relevant to mobile entities and their environments. One area of such research is mobile distributed computing to support and address mobility issues in the wired and non-wired remote environment. This shift in software paradigm has resulted in the birth of the mobile agent which is a new paradigm for distributed application development. The mobile agent paradigm has broken down one of the major barriers in distributed computing which is based on the client/server model. In the client/server model a client needs to establish and maintain a reliable connection with a server(s) in order to communicate although this is less efficient for a highly distributed environment.

Different researchers have defined the mobile agent in different ways. Milojevic (1999) defined the mobile agent as an autonomous software program that can migrate from one platform to another on a heterogeneous network, performing tasks on behalf of the user. Jansen (2002) defined the mobile agent as 'travelling agents', these programs will shuttle their being, code and state, across different resources. Cubaleska and Schneider (2002) defined the mobile agent as a computational process that implements the autonomous communicating functionality of an application. The platform is therefore made up of the computational environment and the agent is also made up of the code and state information that is needed to perform some form of computation (Jansen and Karygiannis, 1999).

In the context of this research, a mobile agent can be defined as an autonomous agent that exhibits mobility characteristics such as persistency, robustness, security assessment for its codes and environment, mobility transparency and fault tolerance. Autonomous agents and multi-agents have become very important in both industrial and academic research. These paradigms draw on concepts from distributed computing, object oriented systems, software engineering, artificial intelligence, economics, game theory, sociology and organisational science. The concept of autonomous agents offers solutions to complex software systems through analysing, designing and implementation (Jennings *et al.*, 1998).

Jennings (2000) identified Agent Oriented Systems Engineering (AOSE) as having the potential to considerably improve the practices of software engineering. It is suggested that concepts of AOSE offer alternative ways of providing software solutions to complex systems. AOSE adopts a multi-agent approach to systems development in an attempt to solve complex problems. A general definition of an agent can be defined based on its general characteristics. Agents are characterised by autonomy, social-ability, interactivity, proactive/goal oriented, reactive, persistent and a desirable property such as mobility, adaptation and rationality (Brustoloni, 1991; Smith *et al.*, 1994; Wooldridge and Jennings, 1995; Franklin and Graesser, 1996; Williams, 2007a). Existing literature about multi-agent systems indicates that autonomous agents are intelligent. In this thesis both autonomous and intelligent agents will be termed agent. Many applications can be created using mobile agents, which means that agents can be integrated to support mobility in several applications. Common applications which utilise mobile agents include remote database searches, information retrieval and messaging applications

which usually carry active and real time content enroute to several remote locations. Since the emergence of the agent paradigm, there has been an influx of different approaches and methodologies to modelling agent systems. This indicates the level of interest that agents have generated in both the commercial and academic research environment. Several researchers have devoted considerable time and effort into developing suitable frameworks and architectures for agents. However, there has been little in the way of providing a systematic practical approach or methodology for developing and addressing mobility issues in mobile agent based systems. There are inherent complexities in using traditional approaches and/or methodologies for engineering agent and mobile agent based systems. These complexities have led to ad-hoc solutions being adopted whereby practical systems have been built from scratch. The overall purpose of this research goal is therefore to develop a methodology using the underlying principles of Genetic Algorithm (GA) approach to modelling mobility in mobile agent based systems to support and overcome some of these shortcomings in current engineering practice and also to improve upon previous attempts.

The rest of this chapter sets the background for the research work undertaken and outlines the aims, objectives and contributions of this thesis. It introduces the methodologies and approaches for developing mobile agent based systems and identifies challenges and shortcomings that need to be overcome to develop the methodology for modelling mobility in mobile agent systems.

## 1.1 Existing Approaches in Mobile Agent-based Systems

With the ever increasing proliferation of new technology owing to the widespread growth in applications, the need to provide a strategy for the analysis, design and deployment of these application systems has increasingly gained a sustained level of interest among agent researchers (Omicini, 2001; Santandiyo *et al.*, 2004; Self and DeLoach, 2004). Attempts which have so far been made by the agent research community have focused on the conceptual modelling of mobile agent-based applications. Methodologies that attempt to address mobility issues are met with notational constraints and implementation considerations. This is as a result of implementing the methodologies as concepts thereby limiting the methodologies (Loukil *et al.*, 2006). There is also lack of consistency in definition for mobile agent in the current approaches that have been developed to model the mobility for mobile agent (Chhetri *et al.*, 2006). Many different approaches to modelling mobility of mobile agent have focused attention on different concepts relating their context specific definition (Belloni and Marcos, 2004). For instance, most existing literature on methodologies and approaches associate the mobility to the agent's role or task which is assigned at any given time during the agent's lifecycle. The role assigned to the agent determines whether it will be stationary or mobile in a multi-agent environment. In a multi-agent system or environment, mobility is viewed as an attribute assigned to an agent or a role performed by the agent and is usually specified in the itinerary of the agent. Meanwhile, the mobile agent as a computational process has been shown to improve the latency and bandwidth usage in distributed applications (Wooldridge *et al.*, 2000). Given this background, it is necessary to develop a methodology to model mobility of the mobile agent.

## 1.2 Research Aim and Objectives

### Aim

The aim of this research is to identify and develop a methodology to model the mobility of mobile agent-based systems. Having this in mind, the objectives of this work are:

### Objectives

- To review methodologies and approaches and to identify deficiencies in current practice.
- To develop a methodology to model mobility in mobile agent-based systems. The methodology will be divided into phases to facilitate the ease in the development process of mobile agent-based systems. The principle of dividing the methodology into phases is to facilitate the development of the software and to make it more manageable.
- To develop criteria for classifying mobility fitness based on mobility specific requirements.
- To develop mathematical models based on mobility requirements identified and derived from the Delphi study and case studies.
- To translate core mobility requirements modelled mathematically into mobility fitness functions solvers for selecting mobility requirements for applications development. The Rastrigin's function will be used for benchmarking performance of the mobility fitness functions.
- To design a layered diagram indicating where and how mobility requirements fit into an online distributed architectural design.

### 1.3 Research Process

As in indicated in Figure 1.1, the research stages covered are as follows; comprehensive and ongoing literature review of multi-agent approaches, methodologies and mobility modelling. Delphi study was employed to solicit the views of experts in software development in the area of online banking and Virtual Learning Environments (VLE) and online games which indicated emerging views and patterns in systems development. Results from this study gave strong pointers to which case studies were appropriate for the methodology development. Case studies were also used for the evaluation and refinement of the methodology which is known as MaMM. The mobility requirements which were derived from both the Delphi study and case studies were simulated and evaluated.

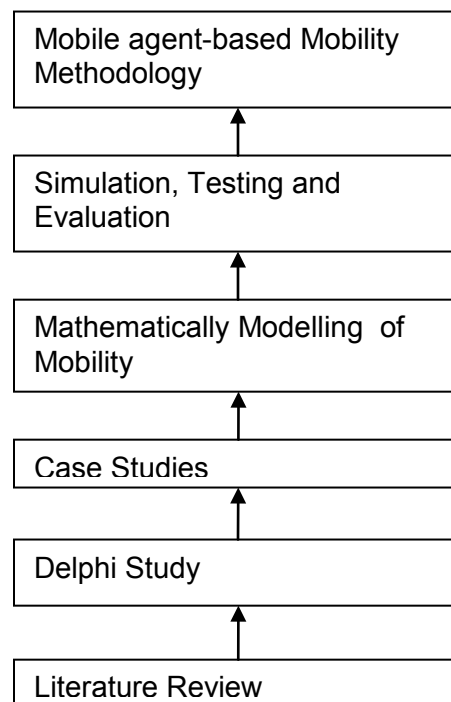


Figure 1.1: Research Process



## 1.4 Thesis Structure

In the rest of the thesis, a thorough and an ongoing literature review is presented. A high level conceptual methodology is presented along with mobility design layer diagram and mobility fitness as captured in the phases of the methodology.

- Chapter 2 reviews the background theory of software agents and agent mobility, the traditional approach to software development processes, the various approaches and methodologies that are available are all described in detail. The limitations of agents are identified against generic requirements for modelling mobility in mobile agent-based systems, which provide the motivation for undertaking this research.
- Chapter 3 describes the research methodology employed in this research. Delphi study and the case studies are discussed and the reasons why these approaches are used. The GA tool used for simulating the results is discussed.
- Chapter 4 presents the key mobility requirements, concepts and mobile agent-based methodology. Concepts and theories that underpin the methodology are discussed. Genetic Algorithms, genetic operators and fitness functions are introduced to the methodology at the analysis phase of the development process.
- Chapter 5 presents results from the modelling and simulation of the Mobile agent-based Mobility Methodology (MAMM). In this chapter, Rastrigins' function is used as a benchmark to measure the effectiveness and performance with the mobility defined fitness function. In analysing the

computational complexity of this approach, an indication is provided that this mobility methodology outperforms other state-of-the-art approaches and methodologies and also explains the reasons why this methodology is better in terms of the mobility and communication requirements for mobile agent-based systems.

- Chapter 6 brings together the conclusions of this research and focuses on the contributions and limitations that the mobility methodology is able to provide. This chapter also outlines future work that can be undertaken to extend the methodology as well as improving the mobility fitness function.

# CHAPTER 2

## Literature Review

### 2.1 Introduction

In this chapter, a literature review of work in the area of multi-agents and mobile agents is presented. In the definition of an agent, mobility is an attribute/characteristic of the agent, this means that to provide a definition for a mobile agent, it has all the properties of an agent including mobility/migration. Researches into multi-agent systems were therefore investigated in order to gain an insight into and extent to which agent mobility has been exploited. Limitations of the existing approaches, methodologies, agent modelling languages, common software development methodologies are highlighted thus serving as a motivation for establishing research objectives for this thesis.

### 2.2 Background Theory

Software agents and Multi-Agents have become very important in research and software development in recent years. Characteristics of software agents include autonomy, pro-activeness, social-ability, re-activeness and mobility. To understand the mobile agent in the context of this research, a historical background of how this has evolved is explained and explored. Software agent concepts have been drawn from distributed computing, object-oriented systems, software engineering, artificial intelligence, economics, sociology, programming and organisational science (Brustoloni, 1991; Smith *et al.*, 1994; Wooldridge and Jennings, 1995; Franklin and Graesser, 1996) .

## 2.3 Definitions of Software Agent

There are several definitions for software agents all of which share similar characteristics (Brustoloni, 1991;Smith *et al.*,1994;Wooldridge and Jennings, 1995; Franklin and Graesser, 1996;Williams, 2007a). In this thesis, software agent is used interchangeably with autonomous agent. The following identifies various definitions provided by different researchers who have been working in this field:

Brustoloni (1991) definition is 'Autonomous agents are systems capable of autonomous, purposeful action in the real world'. He compared his autonomous agents to other agent definitions, and stressed that his agent must of necessity live and act 'in the real world'. He also insists that his agents must be reactive; that is, agents must be able to respond to external, asynchronous stimuli in a timely fashion.

Wooldridge and Jennings (1995) not only provided a definition, but also add an explanation for autonomy, sensing and acting, allowing for a broad, yet clear and concise, range of environments or platforms. Wooldridge and Jennings (1995) defined an agent as '... a hardware or (more usually) software-based computer system that enjoys the following properties:

- Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- Social-ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;

- **Reactivity:** agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
- **Pro-activeness:** agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative’.

An agent can have several characteristics such as the ability to be autonomous, learn, react, be mobile, flexible, and communicative regarding its state, have a continuous running process and be goal oriented. All agents must have at least these characteristics as defined by these requirements; autonomy, reactive, continuous running process and lastly be goal oriented (Franklin and Graesser, 1996).

White (1996) also defined an agent as an entity that occupies a specific place. This entity can move and occupy different places at different times. This independent entity is made up of procedures and the state of an agent. A place is referred to as a network of computers that offer services to any mobile agent that enters it.

These concepts put together offer solutions to complex software systems by analysing, designing and implementing them (Jennings *et al.*, 1998). Whenever an agent is characterised by its ability to move or migrate autonomously in a network, then it is said to be a mobile agent.

Numerous approaches have been developed to model a multi-agent system which is simply a situation of having more than one agent in the network environment with both

stationary and mobile agents (Wooldridge *et al.*, 2000). The design of an agent using different development approaches has been a popular activity in recent years although one of the issues that have been under-developed is mobility in mobile agent-based systems. Most designers assume mobility is part of a role assigned to the agent and concepts to model this mobility have not matured sufficiently enough to support the entire lifecycle of a mobile agent-based system (Wooldridge *et al.*, 2000).

An interesting question which is worth considering before the background theory is examined is why the need for a systematic approach for modelling the mobility in mobile agent-based systems. Mobility issues which surface when modelling mobile agents for migration on a heterogeneous network include the integrity of data, access control, privacy of data and authentication, trust where an agent agrees to meet on a mutually agreed secure host, persistency where a mechanism that permits vital information about the migration activities of the agents to be kept so that a system can resume activities after it has crashed or failed. Since mobile agents are able to migrate from one node to another on a heterogeneous network, performing tasks on behalf of its user without the user's intervention, there is a need to model these mobile agents in order to address the mobility issues as identified. Agent paradigm provides a solution for modelling and implementing complex software system by associating their actions and behaviours to the capabilities of humans.

## 2.4 Existing Agent Modelling Languages

Modelling languages enable software developers to specify requirements of software systems during development processes and also to see the world as made up of software agents. The need to develop/extend existing modelling languages has become important because of the evolving dynamic requirements of agents (Melomey,2007; Melomey *et al.*, 2007a). Most of these modelling languages have been used to express knowledge with respect to goals, tasks and concepts of agents (Melomey *et al.*, 2007a).

The modelling language is a form of communication for modelling purposes. A modelling language guides a developer to clearly represent the internal and external structures and elements that influence agent representation either textually or visually. A visual language allows domain knowledge developers to assemble programs quickly from existing components with its related operations. Visual language offers an added advantage which is to match between the systems to be modelled to the visual abstract. Alternatively, human skills present a higher level of knowledge using textual languages and the associated tool support (Levesque, 1984; Kremer, 1998; Blackwell, 2001).

Unlike the object oriented systems development methodology, AOSE (Kang *et al.*, 2004) has not reached the maturity stage where issues such as modelling languages for system application from the requirement phase through to the implementation of the entire software processes can be established. There is the need for modelling language(s) to model the interaction of agents and their behaviors from the requirement phase through to implementation. Modelling languages are important in order to give a vivid description of agent systems and reasoning about mobility. Issues that often arise

when modelling agent systems are the representation of agents, validating and verifying the agent systems and appropriate model representation.

Formal models with respect to mathematical and linguistic models are essential for describing and reasoning about mobility of agents (Levesque, 1984; Kremer, 1998; Blackwell, 2001). These issues have not, so far, been addressed and appear to have been ignored. However, it is not only the agent's behaviour that need to be modelled but also stakeholders and users, and their interaction with the proposed system modelling languages which should have the ability to capture both internal and external structures of the agent. Even though it is portrayed that the agent has, or of necessity must have, control over its internal structures, there is a need to show the transition from one phase to the other. Control is left entirely to the agent at this point. Modelling for agent systems requires a combination of visual and formal languages. A formal specification tends to provide a solution for some weaknesses of the visual modelling language that may be identified. A formal specification enables models to be defined using precise semantics. Furthermore, to facilitate the transformation from one phase to other, for example from the analysis phase to systems design phase, this requires specialist skill on the part of the program developers and effective communication amongst developers. It is, however, ineffective and inappropriate for communication and discussion with stakeholders (Mauco et al., 2001; Dignum, 2003). Formalising visual languages for conceptual modelling comes with its own set of challenges such as ambiguities of meaning and expression of the graphical notations.

There are quite a number of modelling languages applied to mobile agents and agent systems, most of which draw concepts from the unified modelling languages (Odell et



al., 2000). Table 2.1 provides a summary of existing modelling languages for Multi-agent Systems (MAS) and attempts made in addressing mobility in agent systems (Melomey 2007, Melomey et al., 2007). Criteria used for the comparative analysis in Table 2.1 were derived from distributed system mobility goals (Melomey et al., 2007a). Table 2.1 compares how well the existing modelling languages for MAS;

1. support and model mobility of agent systems
2. model static agent present in an environment and how the agents interact as well as allocate resources upon request
3. provide dynamic modelling support for mobility. That is modeling the sequences of interactions between agents and mobile agents from high level abstraction to low level abstraction
4. preserve the consistency of mobile agents characteristic as it transforms itself through the software process
5. support developers through CASE tool to analyse and design phases of the software process in the systems lifecycle
6. model roles of agents and mobile agents, interfaces and interactions of agents within and outside their environment (external structure modelling).
7. accommodate new and additional words, stereotypes and phrases for mobility adaptation in a dynamic environment or platform (extensible and customisable).

Modelling Language	AUML	AML	SLABS	CAMLE	ASL
Mobility Support	P	P	N	P	N
Dynamic Model Support	P	F	P	P	F
Static Model Support	P	F	P	P	P
CASE Tool Support	N	F	N	F	N
Consistency	P	F	F	F	P
External Structure Modelling	P	F	F	F	P
Extensible and Customisable	P	F	N	N	N

F - Fully Supported  
P- Partially Supported  
N – Not Supported

Table 2.1: Analysis of Existing Modelling Languages for MAS

Results from Table 2.1 indicate the strength and weakness of each modelling language for Multi-agent Systems (MAS). The following sub section will present each language in detail with their strengths and weaknesses.

### 2.4.1 Agent Unified Modelling Language

Agent Unified Modelling Language (AUML) is an extension to the unified modelling language. There are no restrictions to the extensions one can make to UML. For example, Mouratidis et al. (2003) provided extensions on deployment and activity diagrams to model agent mobility in Tropos. Similarly, another approach in modelling mobility was the extension of activity diagrams using UML 1.5 (Kang *et al.*, 2003).

## 2.4.2 Agent Modelling Language

Agent Modelling Language (AML) has features for capturing multi-agent systems. AML combines both visual and formal language for modelling and agent specification, and it draws its concepts from multi-agent systems theory (Cervenka *et al.*, 2004). AML also specifies models and document systems by extending UML 2.0. AML according to Cervenka *et al.* (2004) is a semi visual modelling language based on the concepts of multi-agent systems and also specifies the models and documents a system. It reuses concepts from UML and also makes use of mechanisms for specifying and extending UML-based languages. It is also easy to incorporate into UML based CASE tools. The language syntax, semantics and notations are defined at AML meta-model and notation level (Trencansky and Cervenka, 2005). AML therefore provides constructs for modelling applications which represent and exhibit characteristics in multi-agent system.

### Strengths of Agent Modelling Language

The strength of AML is in capturing multi-agent systems using UML 2.0 to model agent specification. AML has the ability to easily incorporate UML 2.0 to existing UML case tools and to model static mobility and the agent execution environment.

### Limitations of Agent Modelling Language

.AML does not model dynamic mobility. AML only models mobility through the design phase. There were no construct or mobility supports to model mobility of the mobile agent.

AML focus is on the development of multi-agent systems and not on mobile agent systems (Cervenka and Trencansky, 2004; Cervenka and Trencansky, 2007). According to Cervenka and Trencansky (2007) ‘...AML provides a rich set of modeling constructs

for modeling applications that embody and/or exhibits characteristics of multi-agent systems' and also '.... how entities can get to a particular node of the physical infrastructure'. AML therefore provided constructs to model multi-agent systems only.

### **2.4.3 Specification Language for Agent-Based Systems**

Specification Language for Agent-Based Systems (SLABS) provides the developer with language facilities together with features for formal specification as well as the verification of agent based systems (Zhu, 2001). Its focus is geared towards the development of large scale complex systems. According to Zhu (2001) SLABS is based on a generalised model of agents rather than a specific agent theory, is decomposable and integrates new concepts such as caste and provides language facilities for AOSE.

### **2.4.4 Caste-Centric Agent Modelling Language and Environment**

A Caste-Centric Agent Modelling Language and Environment (CAMLE) is a language based on the notion of caste and draws on the concepts of SLABS (Shan and Zhu, 2004). Caste by definition is a set of agents with the same behaviour and structure. SLABS combine both graphical modelling with formal specification language by automation. CAMLE introduced visual models at the design stage of the development process which are caste, collaboration and behavioural. Diagrams in the caste model specify relationships including their movement from one caste to the other (Shan and Zhu, 2004; Zhu and Shan, 2005). Collaboration models include diagrams organised in a hierarchical order depicting the interaction of agents and their relationship in the system. Finally, the behavioural model diagrams define how agents decide on what action to

take and how it changes states depending on a given scenario. All these models come with well defined associated notations.

### **2.4.5 Autonomy Specification Language**

Autonomy Specification Language (ASL) is a language that allows for an exact specification of activities which will be carried out by group of agents, deontic constraints which place an imposition on these agents and the implications brought about by executing activities on a specific constraint (Weib *et al.*, 2006). ASL has its strength in the operational modelling for specifying the autonomy of the agent. An ASL concept defines roles through a set of activities as well as specifying the behaviours that it conforms to or deviates from accepted norms of agent systems. In specifying the behaviours, it enables behavioural prediction of agents through the roles they assume. Furthermore, ASL enables software designers to specify the autonomy of agent as well as allowing the detection and resolution of induced conflicts that occur during runtime. Weib *et al.* (2005) argues that to be able to implement autonomy in a commercial, scientific and industrial application, it can only be achieved through a systematic process of rigorous modelling and verification. This will offer a high level of dependability on systems that can be granted permission to act autonomously. Without this kind of dependability, it will be difficult for agents to be used in the ecommerce, industrial and scientific applications. ASL has an operational character which is expressive and also flexible with reference to the autonomy of an agent (Weib *et al.*, 2006).

### Limitations of Autonomy Specification Language

ASL constructs for modelling and assigning individual agents for a role has not yet been explicitly defined or expressed and no mention were made of how mobility issues will be handled (Weib *et al.*, 2006). Weib *et al.* (2006, p.1) introduced ASL as a formal language *'...that allows for a precise specification of the activities to be carried out by a set of agents, the deontic constraints imposed on these activities, and the implications of activity execution on particular constraints (i.e. constraint dynamics)'*.

### **2.4.6 Other Extensions to UML**

There have been attempts to model mobility of the mobile agent by extending UML 2.0 which is known as Agent UML (AUML). The AUML extensions are activity and deployment diagrams.

AUML was proposed as an extension to UML to be used as a tool to model communication protocols and interactions in multi agent system (Bauer, 1999; Odell 2000, Bauer *et al.*, 2001). AUML has been used by some agent researchers to model the extension of activity and deployment diagrams. These extensions model the static views of the mobile agents rather than the dynamic view. Other languages that have been used to model mobility are Agent Specification Language (ASL) (Weib *et al.*, 2005) and Agent-based Modeling language (AML). AML provides the definition for meta-classes that are used to model the structure and behaviour of mobility of entities (Cervenka and Trencansky, 2004).

- Baumeister et al. (2003) extended UML 2.0 to the AUML activity diagram in an attempt to model mobility. The authors introduced new stereotypes such as mobile, mobile location, clone and move to model mobility. Baumeister et al. (2003) also introduced new concepts such as mobile objects, locations, nested locations, actions and two notional variants. With the introduction of new stereotypes and concepts, the authors attempted to answer the question; who is performing an action and *where* the action is being performed. The authors used swimlanes to represent objects to indicate who is performing an action and the object's mobility with respect to the object's location.

#### Limitations of Other Extensions to UML

The AUML extensions made by Baumeister et al. (2003) provided a concept of nested locations but this was not properly defined and illustrated. The mobile location stereotype also lacked clarity. The AUML activity diagrams made by Baumeister et al. (2003) were meant to model mobility, however, the stereotypes and concepts do not have a direct bearing on agents nor mobile agents but rather objects and their mobility relationships.

### **2.4.7 AUML Deployment Diagram**

The deployment diagram in UML provides the physical resources in the system which includes the connections, computers or nodes and components.

The proposal by Mouratidis *et al.* (2002) was to provide definition for origin, destination and mobility path as an extension to deployment and activity diagram in order to model mobility based on UML 2.0 meta-structures. The origin is the platform where the mobile agent begins its execution and the destination is the platform where the mobile agent finishes its execution. Mobility path is the path between the origin and destination. Mouratidis *et al.* (2002) also provided notations for capturing mobility of agents in the network.

#### **2.4.8 AUML Activity Diagram Extensions**

An activity diagram in UML demonstrates the dynamism of a system. This was achieved by modelling the flow of activity. An activity is a representation of an operation in a class of a system that results in changes in the system state.

The extension by Mouratidis *et al.* (2002) was based on UML 2.0 meta-structures for the activity diagram which captures the sequence, concurrency and iteration of the mobile agent. It provides answers to how it is able to get to its intended destination. The extended activity diagram provides concepts that capture the sequence of movement, mobility path details and decisions an agent makes regarding which path it should take (Mouratidis *et al.*, 2002). Diamond notations of UML were also used to capture situations where a mobile agent has to decide which node to visit from the available options. These knowledge statements are then converted to codes and added to the knowledge database of the mobile agent during the implementation stage.



Poggi *et al.* (2004) also extended the deployment and activity diagrams in an attempt to model mobility. The authors introduced concepts and notations such as home, mobility path, and visitor. They also introduced dotted lines to represent messages and dashed lines towards platforms where a mobile agent may visit. The activity diagram of UML was also extended by introducing concepts such as bounce failure and return path which indicate two statements with two arguments (Poggi *et al.*, 2004). AUML is also able to model the sequence of activities, concurrency and iteration of the movement of the mobile agent.

#### Limitations of AUML Activity Diagram Extensions

The AUML deployment and activity diagram introduces additional concepts and notations which model static mobility and dynamic mobility for agent-oriented software development (Mouratidis *et al.*, 2002; Poggi *et al.*, 2004). However, AUML activity and deployment diagrams were not able to model the dynamic mobility of the agent from one node to the other (Mouratidis *et al.*, 2002; Poggi *et al.*, 2004). The extensions were able to model only the static mobility (Mouratidis *et al.*, 2002; Poggi *et al.*, 2004). The AUML activity diagram did not demonstrate the continuous established link whereby a mobile agent can make an independent decision (Mouratidis *et al.*, 2002; Poggi *et al.*, 2004). Also there was no mention of any form of itinerary for the mobile agent which is central to the development of an internal structure for the mobile agent. For example, Kosiuczenko (2003, p.1) used sequence diagram to model mobility of an object and noted that '*There are several kinds of UML diagrams for convenient modelling of behaviour, but these diagrams can hardly be used for modelling mobility*' and then introduced extension to model interaction of mobile objects by proposing '*... a new graphical notation for modelling interaction of mobile objects*'. Kang *et al.* (2004, p.5)

concluded by pointing out '*...that our interpretation of UML 2.0 Activity diagrams is based on mobile calculus...*' Poggi *et al.*,(2004, p.14) also proposed '*...an AUML deployment diagram, that is an UML deployment diagram enhanced with agent based concepts....*'.

## 2.5 Existing Multi-Agent Approaches

The approaches available for modelling the mobility of mobile agents are predominantly an extension to the Unified Modelling Language (UML) diagram. UML provides unification and formalisation for methods of the numerous approaches to the object oriented (OO) software systems lifecycle (Jacobson *et al.*, 1998). UML is a specification language for object modelling and a general purpose modelling language which includes a standardised graphical notation used to create an abstract model of a system (Jacobson *et al.*, 1998). UML is made up of the following diagrams: use case, class, sequence, collaboration, package and components diagrams.

The following section provides an overview of MaSE, GAIA, TROPOS and Prometheus methodologies. Table 2.2 also provides a summary of existing Multi-agent Systems (MAS) methodologies and their strengths in the phases of systems development.

Methodology Phases	Requirement Phase	Analysis Phase	Design Phase	Implementation Phase
MaSE	No	Yes	No	No
GAIA	No	Yes	Yes	No
TROPOS	Yes	No	No	No
Prometheus	No	Yes	Yes	Yes

**YES** – Strength of the Methodology    **NO** – Limitation of the Methodology

Table 2.2: Existing Multi-agent Systems Methodologies

### 2.5.1 Multi-agent Systems Engineering (MaSE) Methodology

DeLoach *et al.*(2001) developed a Multi-agent Systems Engineering (MaSE) Methodology. The approach used was to add the move command in the MaSE analysis models with its associated transformation requirements and was incorporated in the design functionality (Self and DeLoach ,2004). Design models were further translated into java based agents that operate within a mobile agent environment. DeLoach (2004) also introduced dynamic agents with one of the following properties:

Cloning: this is the ability of an agent to create a replica or an instance of itself either at the same location or at different locations DeLoach *et al* (2001).

Instantiation: an agent having the ability to create instances of another class other than itself.

Mobility: the ability of an agent to migrate from one node to another.

In MaSE, only the analysis and design phases of mobile multi-agent systems were considered based on the following assumptions:

1. The agent determines when to move even though another agent or the agent platform may advise the agent's when to move.
2. The actual movement of an agent is handled by the appropriate mobile agent's platform protocol which is similar to FIPA's Simple Migration Protocol (SMP). The protocol with a request is sent to its mobile agent platform which terminates the agent. It is then sent to the destination platform where it is restarted. The platform is responsible for the movement and the agent is responsible for restarting itself in an appropriate state.

MaSE methodology has two phases and seven steps (DeLoach, 2004). The author's focus was on the output models of the analysis phase i.e. role models and concurrent tasks. In other words, the analysis phase defines a set of roles to be played by the agent as well as a set of tasks that also define the behaviour of specific roles and lastly a set of coordination protocols between those roles.

According to DeLoach (2004) a move activity within the state of the concurrent task diagram returns two values which are Boolean value and a reason value. The Boolean variable always returns either a success or failure. The reason value provides a reason why there is a failure or a success. For example if the reason value is failure, it provides the reason why a move failed and also provide the agent with knowledge to recover successfully from failure.

Basically all the tasks from the analysis phase are translated into various components in the design phase. This phase is where the internal agent architecture is defined. . Apart from requesting <move>, each component of the mobile agent is designed to respond to a <move> that is by being able to save its internal state and to restart from its new location(Self and DeLoach,2003). Every agent has a component that is created to oversee the operations and provide interaction between the components. The component for the mobile agents should have the capacity to transform itself in order to handle shutdown and re-initialisation of all the agent components.

Agent class in MaSE methodology is a model for the different types of agents in a system (DeLoach *et al.*, 2001). It is similar to the object class as in the object oriented paradigm. In this particular case an agent class is defined by the role it plays in the systems and not by attributes and methods as in the object oriented paradigm. Tasks that are associated with a role automatically become a component of the agent class depending on its role in the system (Self and DeLoach, 2003). The agent component is responsible for completing most of the agent mobility function and actually determines whether the agent should move or not after a move request is made.

### Limitations of MaSE

The MASE methodology focused on the output models of the analysis phase of systems development and failed to identify why mobility is needed and its association with the requirements of the systems. Wood and DeLoach (2000, p.208) stated that '*...the methodology does not consider dynamic systems where agents can be created, destroyed, or moved during execution*'.

### 2.5.2 GAIA Methodology

Generic Architecture for Information Availability (GAIA) is a methodology for agent oriented analysis and design (Wooldridge et al., 2000; Zambonelli et al., 2003). It is a general methodology that can be applied in many phases of multi-agent systems. According to Wooldridge *et al.* (2000) it handles both micro and macro level aspects of the entire system.

GAIA methodology has been identified as suitable in large scale commercial applications by the authors Wooldridge *et al.* (2000) and Juan *et al.*, (2002), Zambonelli *et al.*(2003). According to Wooldridge *et al.* (2000) agents in a GAIA system should have at least 100 agents in typical applications. In this methodology, the requirements of the system are independent which allows the analyst to adopt a systematic approach from the requirement phase to the analysis phase.

GAIA borrows its terminology and notations from object oriented analysis and design. GAIA is intended to help software engineers in understanding and modelling of complex systems. The formal notation for the expression of permission in GAIA methodology is based on FUSION notation for operation of schemata.

#### Strengths of GAIA Methodology

GAIA provides an approach for developing collaborative multi-agent systems providing models for static interactions, services and interactions in a given environment covering

the analysis and design phases of systems development (Wooldridge *et al.* 2000; Juan *et al.*, 2002; Zambonelli *et al.* 2003).

#### Limitations of GAIA Methodology

GAIA lacks the concepts and graphical notations to support the modelling and reasoning of the agents' mobility and the social interaction in an environment or platform. Huang *et al.* (2007) identified these limitations in the GAIA design phase for which attempts were made to provide extensions to designing and developing agent-based software. Wooldridge *et al.* (2000, p.24) also cited that '*There are several issues remaining for future work.....the representation of inter-agent cooperation protocols within Gaia is currently somewhat impoverished....we will need to provide a much richer protocol specification framework*'.

### **2.5.3 TROPOS Approach**

TROPOS is a requirements-driven methodology (Perini *et al.*, 2002; Castro *et al.*, 2002). It was developed to provide support for all the analysis and design activities during the entire software development process. TROPOS covers the early and late requirement phases, as well as the architectural design and implementation phases. It makes use of actors, goals and actor dependencies. Bresciani *et al.* (2004) defined TROPOS as a software methodology which allows the exploitation of the flexibility that is provided by agent oriented programming. Agent oriented programming encourages the need to accommodate open architectures that changes continuously and dynamically i.e. evolution of new requirements and new components.

### Strengths of TROPOS Approach

The greatest strength of Tropos approach lies in its identification of early and late requirements for the system in the requirement phase of the systems development process (Perini et al., 2002; Castro et al., 2002; Bresciani *et al.*, 2004).

### Limitations of TROPOS Approach

TROPOS was developed for multi-agent system applications and not for mobile agent system application development according to Garzetti et al.(2002) and Bresciani *et al.* (2004), hence it failed to provide the necessary processes and concepts to model dynamic mobility for systems development. For instance , Castro *et al.*, (2002, p.365) explained that Tropos include the following ‘...*methodology includes techniques for generating an implementation from Tropos detailed design. Using agent-oriented programming platform for implementation seems natural, given that the detailed design is defined in terms of (system) actors, goals and inter-dependencies among them*’. The explanation did not include mobility of an agent.

## **2.5.4 Prometheus Methodology**

Prometheus methodology is a more detailed process for specifying, designing, and the implementation of intelligent agent systems (Padgham and Winikoff, 2002; Padgham *et al.*, 2005). The goal of this methodology is to have well defined deliverables which are practical enough to be used by those who do not have an exclusive knowledge of agents to be able to develop intelligent agent systems.



Padgham and Winikoff (2004) stated that Prometheus methodology is a software methodology that is able to aid in the transition of agents from research laboratories to industrial practice. It distinguishes itself from other methodologies by the following features;

- provides detailed guidance on how each process should be carried out in Prometheus.
- provides supports on the design of goal oriented agents and agents that have plans
- gives coverage on activities from requirements specification of agents through to detailed designs
- facilitates support tools in the form of Prometheus Design Tools (PDT) which are available on the internet.
- aimed at these two markets (education and industrial developers), this methodology was successfully implemented and was given positive feedback and comments which was used to improve the methodology.

Padgham and Winikoff (2004) stressed that the Prometheus methodology is a general purpose approach and that for the detailed stages it is allowed to make certain assumptions.

The Prometheus methodology is defined as concepts, notations for capturing design and also a technique that provides guidance on how to carry out steps in the processes. The method has three main phases which are: the systems specification; the architectural design; and the detailed design.

### Limitations of Prometheus Methodology

Padgham and Winikoff (2005) stressed that the Prometheus methodology is a general purpose approach for developing and implementing intelligent systems and therefore does not provide notations and concepts for modeling mobility. Fletcher (2007, p.342) concluded that *'the key conclusion is that this methodology is very suitable for developing static interactions between agents...'*

## **2.6 Mobile Agent**

A mobile agent is an autonomous software program that can migrate from one platform to another on a heterogeneous network performing task on behalf of the user (Milojicic, 1999). It is a computational process that implements the autonomous, communicating functionality of an application and is able to migrate from one computer to another over a network. The platform is made up of the computational environment and the agent is also made up of the code and state information that is required to perform some form of computation (Cubaleska and Schneider, 2002). In other words, the platform provides a physical environment for the deployment of agents and agent can be said to have a set of attributes called state which describe its characteristics. Agents communicate using an Agent Communication Language (ACL).

Jansen (2002) defined a mobile agent as 'traveling agents', and these programs will migrate their being, code and state, among resources.

A mobile agent has been defined in several ways by a number of authors, some of which are presented as follows:

- Mobile agents are software abstractions that can migrate across the network representing users in the various tasks as defined by Milojevic *et al* (1999).
- Jansen and Karygiannis (1999) defined a mobile agent as a computational process that implements the autonomous, communicating functionality of an application and is able to migrate from one computer to the other over a network. This means that mobile agents are software agents that possess the ability to move from one host to the other. A host platform may consist of more than one agent. The platform is made up of the computational environment and the agent is made up of the code and state information that is required to perform some form of computation. The platform provides the physical environment for deployment by the agent. The number of mobile agents required depends upon the size and type of application.

A mobile agent is therefore characterised by its ability to migrate from one host to another during execution. It is important to note, that its migration is not always from one host to another host but from any place or location that will allow it to resume its execution.

Mobile agents' architectures have contributed to the solution of problems caused by unreliable network connections, reduction of network loads and latency that is sending agents to where data resides on networks and thereby reducing network bandwidth

consumption to a minimum. The ability of Mobile agents' to sense their environment and react dynamically to changes in the environment makes them very useful in the area of intrusion detection. Mobile agents have been used in industrial and commercial applications for example ecommerce, manufacturing, air traffic control, and real time system and information management.

In this research the mobile agent is defined as autonomous agent that exhibits mobility characteristics such as persistency, robustness, security assessment for its codes and environment, mobility transparency and fault tolerance. Furthermore a mobile agent is defined as a program that exhibits persistency, fault tolerance, synchronisation, remote addressing and referencing, calling, invocation, execution, and remote code execution and migration capabilities.

The mobile agent can be implemented using one of the following two technologies; remote objects (Vinoski, 1997) or mobile code (Baldi *et al.* 1997). An example of remote objects implementation is Aglets (Lange, 1997). An example of a mobile code implementation is Telescript (White, 1996) and AgentTCL (Gray, 1995). There are other Java-based mobile systems such as JADE (Java Agent Development Framework), Aglets, Concordia and Voyager. JADE was developed by Telecom Italia and is controlled via a remote graphical user interface and is available on the JADE website.

There are many benefits to be derived from the nature of distributed computing of agent for which mobile agent plays a central role in performing task related to it. The benefits are:

- Distributed knowledge expertise. There are times that knowledge required to solve some type of problems may not reside in a central or single resource, mobile agents are required to migrate to other environment or platforms in search of solution where stationary agents are limited by resources.
- The nature of mobile agents in a distributed system allows for system modularity as opposed to a monolithic program. Large complex systems are broken down to smaller adaptive and proactive modules.
- Improve speed. Due to the parallelism which is a natural outcome of modularity, agents and mobile agents have their own local memories and processor.
- Modularity in mobile agent-based systems allows for efficiency in that only a part of resource are used for providing solution for a problem.

There is currently no methodology for modelling the mobility of mobile agent other than using existing multi-agent approaches to capture mobility as a role in multi-agent development depending on the ad-hoc tasks assigned to an agent when needed (DeLoach *et al.*, 2001; Wooldridge *et al.*, 2000; Zambonelli *et al.*, 2003; Perini *et al.*, 2002; Castro *et al.*, 2002; Padgham and Winikoff, 2002; Padgham *et al.*, 2005).

## 2.7 Traditional Software Development Process Models

A popular and acceptable definition for a software development methodology is the collection of processes, procedures, standards and policies used by a software development team to practice software engineering in order to meet a particular

requirement. Somerville (2001) describes many of such software process models in his book on software engineering. There are several steps procedures and activities involved in systems development which include the waterfall model, rapid prototyping model, spiral model, the evolutionary model and incremental model. There are also agile and rapid applications development models.

There are several benefits why iterative and incremental techniques were chosen as part of MaMM. They are:

- It helps to alleviate against earlier risk that might stem from architecture or integration issues
- Allows for the delivery of an independent module to be implemented and executed incrementally
- Allows progress to be monitored with the detection of problems being identified and isolated.

### **2.7.1 Waterfall Model**

This is a classic model introduced in the 1970's by Winston W. Royce. The waterfall model is usually modelled in cascade which begins with the establishment of a specific requirement, followed by the design, implementation, system testing and finally the release to customer (Royce, 1970). There is no iterative feature in the waterfall model and it works well when the requirements are known and can be address smoothly without having to go back over the previous steps.

### **2.7.2 Rapid Prototyping Model**

This is commonly known as a 'throwaway' or 'operational' model. The development process for the Rapid Prototyping Model (RPM) produces a program that performs an important set of functions in the final software product (Isensee and Rudd, 1966). The approach is mainly used to test the implementation method, a specific language or user acceptance of a product. If successful, it serves as the basis for actual product development after which the prototype is then thrown away or discarded (Andrews, 1991). The difference between the waterfall model and the rapid prototyping model is the speed in which the system is developed.

### **2.7.3 Evolutionary and Iterative Model**

The evolutionary model offers a continuous feedback loop between each phase of the software life cycle or systems development (Greer and Ruhe, 2004). The iterative method is used incrementally thereby producing an executable release of the software product. Developers and designers usually apply both the incremental and iterative model effectively in procedural and object oriented programming.

### **2.7.4 Incremental Model**

The purpose of this model is to develop a fully operational and quality system at each development phase. The model is build and implemented in an incremental fashion with various components of the system developed at different times and integrated as a complete system when all the components are finished (Qui and

Riesbeck, 2008). The advantage this method offers to the developer/designer is the ability to break down complex tasks into smaller manageable components.

### **2.7.5 Spiral Model**

The spiral model was developed by Boehm (1988) and was developed as an enhancement to the waterfall model. The spiral model adds a preceding risk analysis to each cascade of the waterfall model and spiral model is mainly used in large scale software development. This model has proved to be successful where the goal of the system is software reuse and specific software objectives can be incorporated.

## **2.8 Discussion**

In developing applications where it is advantageous to apply agent and mobile agent concepts as part of the solution, it is imperative to ensure a high level of cohesion of agents, mobility concepts and models through the development process which means that the coupling among these elements in the systems is kept to a minimum. Complex mappings of agent concepts from the analysis phase to the design phase needs to be kept at minimum. When this is achieved it enhances agent understandability, traceability and maintainability. To represent real world agents, they may be assigned with roles of which some may need to be mobile. Collectively, all these function represented influence the behaviour of the agent making decisions about mobile agents in a specific system regarding the migration to and from a platform, the following requirements must be taken into consideration for a robust solution; reliability; security; performance; fault tolerance; and transparency. To address some of these issues, concepts for mapping agent, describing agent's behaviour, dealing with communications, specifying and constraining agent migration are required. Also concepts for proving patterns to help designers



achieve transparency and other features must be considered. These issues are fundamental in an open distributed system environment.

## **2.9 Summary**

In this chapter, current literature about agent, modelling languages, agents approaches and methodologies, common software development methodologies, mobile agent and mobility concepts have been discussed. This has indicated some weaknesses and limitations in the lack of a methodology to model mobility in mobile agent based systems. These limitations make the existing approaches and methodologies inadequate for modelling mobility in mobile agent based systems. In the light of such inadequacies, chapter 3 will present the research methodology which justifies the choices and methods made in this research.

# CHAPTER 3

## Research Methodology

### 3.1 Introduction

This study was conducted to gather information for the development of a methodology to model and build mobile agent-based systems. The Delphi study was used to collect expert views on emerging views and software practices in mobile agent software development. The outcome of the Delphi study and the review of existing literature gave a focus on the selection of the three case studies which exhibit the characteristic requirements of mobile agents and the commonalities that exist among the requirements for developing applications with a high level of migration in their specific environment. Case studies have been used in research as a technique which is made up of thorough investigation in an effort to provide in depth analysis of the processes being studied (Yin, 1994). Case studies can be used together with other data collection methods (Yin, 1994). The case studies were used to collect further details from experts and senior analysts who contributed their individual perspective of how and what processes were involved in the different types of online projects they were involved in when they were interviewed. These provided greater insights into the dynamics of online systems development. Mobility requirements identified were modelled and fitness functions were formulated. The modelled mobility requirements were assigned fitness function for evaluation and accuracy and simulated using a GA tool in the Matlab environment. Simulation was suitable for this research because of the benefits it offers in solving complex problems in systems development.

## 3.2 Delphi Technique/Study

The Delphi study was used in this thesis to solicit converging opinions (Dalkey and Helmer, 1963; Linstone and Turoff, 1975; Hsu and Sanford 2007) such as emerging trends, view from experts and software practices in mobile agent development as in generating a consensus from an expert.

### 3.2.1 The Delphi Process (Number of Rounds)

Three rounds of questionnaires were used in this thesis. A number of early Delphi study researchers accepted and agree that three rounds is enough to reach consensus from experts in most cases (Cyphert and Grant, 1971; Brooks, 1979; Ludwig, 1997; Custer *et al.*, 1999).

### 3.2.2 Selection of Experts

According to existing literature, choosing appropriate experts for the study has a direct correlation to the quality of outcome of the results and therefore the selection should be dependent on expertise in the area of discipline (Judd, 1972; Taylor and Judd, 1989). According to Delbecq *et al.*, (1975), participants qualified to take part in a Delphi study falls into three categories the decision makers in the top management who are likely to make use of the results of the study; professional members and support team; and finally, the respondents whose expertise are being sought after. The experts who participated in this Delphi study were selected based on their expertise in projects they have led and been involved in and their software practices for the types of project they have led. The experts who participated were drawn from the National Health Trust (NHS) UK, Barclays Bank Plc UK, HSBC Bank Plc UK, NatWest Bank UK and software

developers with over 15 years experience in leading diverse software projects in the United States of America. Delphi studies have been used in a number of research areas to identify and forecast on research issues. The number of respondent varies from 4 to 171 according to existing literature which is modifiable to suit research issues for which expert views are being sought (Gustafson et al., 1973; Robertson et al., 2005). The size of experts used for this study was seven which falls within the size needed for Delphi study.

### **3.2.3 Data Collection Process**

Once the experts were selected, three Delphi rounds were use to collect data from them. Responses indicated which case studies were suitable for developing the methodology. As in most Delphi studies the first round was open-end questions/issues which served as a backbone for establishing the trends and software practices (Custer *et al.*, 1999).

#### **3.2.3.1 Round 1**

The first round of the Delphi phrased the potential strengths, weakness, opportunities and threats into issues pertaining to methodologies for modelling mobility of mobile agent-based systems. The following were the issues that opinions were solicited on:

##### Issue 1

Do you follow any particular development lifecycle? (e.g. tradition method of software development, Agile or other).

##### Issue 2

What qualities, in order of importance are critical for successful development and implementation of online applications? (Qualities such as synchronisation, security, concurrency, resilience/persistency, remote messaging, availability or others).

#### Issue 3

What are your preferences, if any, for methodology, modelling languages and programming languages?

#### Issue 4

What development approach do you use (object oriented, agent oriented, and mobile agent oriented or other)?

#### Issue 5

Are there any emerging issues in the development of online banking applications?

#### Issue 6

What can be done to improve existing mobile agent technologies/mobile technologies and methodologies?

The responses gathered from the experts were subjective based on the judgments and expertise of each. Their comments were analysed based on the strengths and weaknesses of methodologies, qualities and features for the development and implementation of online applications.

### **3.2.3.2 Round 2**

Based on the responses of the experts from the first round of the Delphi, new and emerging issues arose which needs further clarifications. These were further rephrased and sent to different experts to respond to. The second round sought to clarify the non-

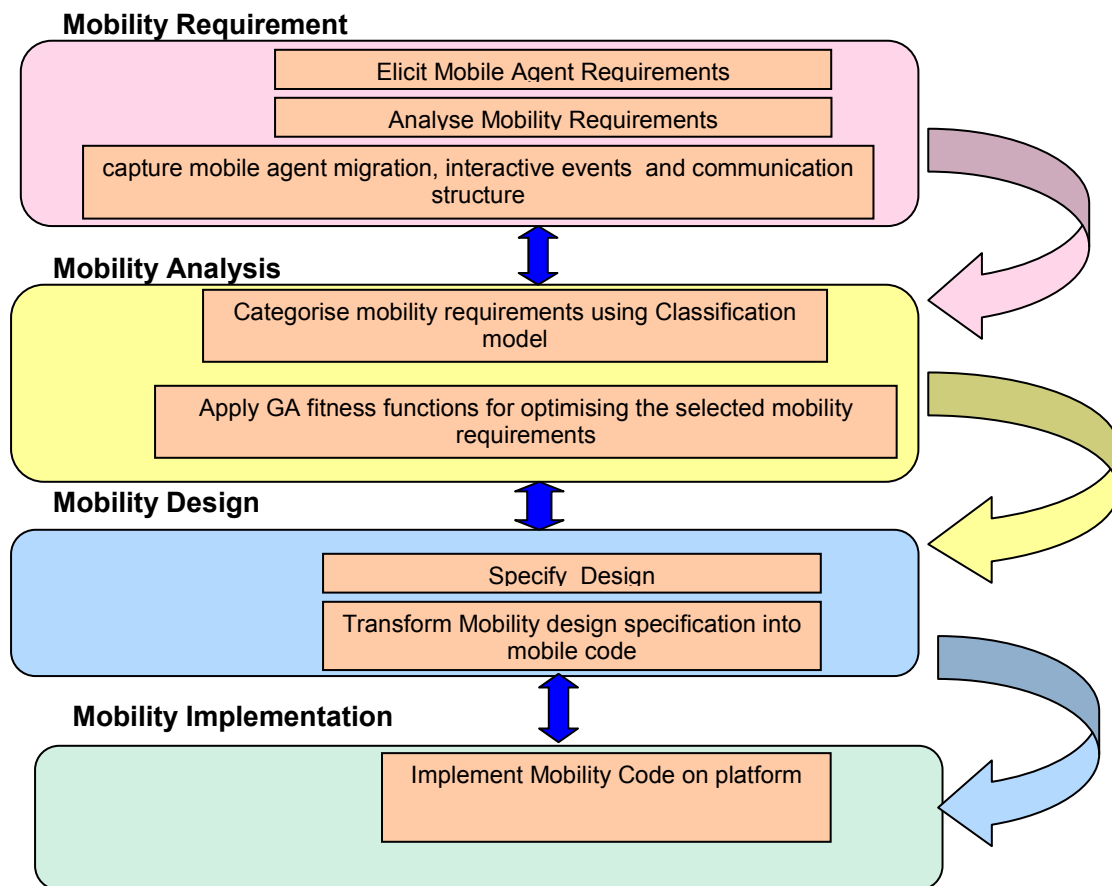
functional requirement order of importance, functionality issues in modelling, programming and testing for mobility in mobile agent-based using a methodology.

### Issue 1

In order of importance how would you rank the following non-functional requirements; synchronisation, remote method invocation, availability and migration, scalability?

### Issue 2

Are the methodology phases adaptable for online application such as on line banking, gaming and Virtual learning environments? Please see Figure below and comment?



Issue 3

In term of back end integration, can the development phases be integrated with other systems?

Issue 4

What are the critical functionality issues and challenges in programming for mobility?

Issue 5

List 3 main functions of agent software in online applications such as online banking?

Issue 6a

How will you test the functions of a mobile agent?

Issue 6b

What data is essential for testing a mobile agent?

**3.2.3.3 Round 3**

The third round was based on the responses from the second round which were refined and sent back to the experts for evaluation and modification of their responses. A draft of MaMM phases was included for them to evaluate. The MaMM phases were results of gaps found in the knowledge after reviewing literature, two rounds of Delphi and case studies selected which stem from the responses on the Delphi study. See Appendix B for issues emailed to experts and their responses.

### 3.2.4 Analysis of Delphi Study Data

Data collected from the responses were analysed which gave an indication of the importance of selecting cases that exhibit the characteristics of autonomy, migration, availability synchronisation and security. In this research, the focus was on mobility and not on the security issues.

Some of the mobility requirements for developing mobile agent-based systems were derived from the case studies introduced in this chapter. These mobility requirements identified and derived from both the Delphi study (Appendix B) and case studies (Appendix C) focused on open-ended structured issues in online application development and formed the basis for MaMM development. The use of more than one case scenario and the analysis from the Delphi study provided the basis for developing a generalised mobility methodology.

The analyses of Delphi study highlighted the following:

- Use of variant approaches to systems development. Software development houses use a variant of the traditional approach to systems application development and the more contemporary approach such as Agile development practices and Rapid Application Development (RAD) were dependent on the scale of the project (Appendix B, Expert Response 1, Expert Response 2). For example in large projects, development teams tend to depend more on the variant of traditional approach while small projects leans more on Agile methodology.
- Order of importance of the mobility specific requirements. Emphasis was placed on the order of importance of mobility requirements critical to the



development and implementation of mobility specific application. The order is as follows; security, availability and migration, persistency/resilience, synchronisation, remote method invocation and reliability as indicated by Figure 3.4.1 given the number of experts who participated in the study (Appendix B: Expert Response 1, Expert Response 2, Expert Response 3, Expert Response 4). For example in online banking, Appendix B: Expert Response 2 indicates the order of importance to be security, availability and reliability. The reason being that in online banking, banks need to be seen as doing the right things to safeguard against reputational damages, fraud and the threat of regulatory sanctions. This order is applicable to all ecommerce websites as well. On the other hand Appendix B: Expert Response 3 has the order of importance as availability and migration, scalability, remote method invocation and synchronisation.

- Challenges in software development. Some of the challenges in the software development such as coding of methodology, language support for the methodology, coding and programming style were indicated and/or highlighted in Appendix B: Expert Response 2.
- Adaptability of some of the phases of MaMM draft. Responses indicated that some phases are adaptable but not all due to diversity in software applications and their specific requirements (Appendix B: Expert Response 3, Expert Response 4).
- Potential of integration with other existing systems. The MaMM has the potential of being integrated with other existing systems. When experts' views were sought on the three most important functions of agent software in online applications, they indicated the following; reduction of development time,

minimisation of human error and cost effectiveness (Appendix B: Expert Response 3, Expert Response 4).

### **3.2.5 Confidentiality of Delphi Study**

The identity of all participants who responded to issues were treated with utmost confidentiality as in all Delphi studies.

## **3.3 Case Studies**

According to authors when multiple cases are selected for studies the cases must complement each other which tend to make the studies more robust in order to make a significant contribution (Herriott and Firestone, 1983; Yin, 1994). In this research three different cases were considered which complement each other. The various data collection methods for case studies are observation, interviews, documentary evidence and participant observation. The method employed in this research for data collection was observation. This observation method allows first witness accounts in the collection of events as it unfolds (Vyas and Woodside, 1984). The senior systems analysts and consultants who contributed to these cases were selected from the banking sector, online gaming and VLE developers.

### **3.3.1 Data Collection and Analysis Process**

The cases were studied by interviewing senior systems analysts and consultants. Seven of them granted interviews, including two recorded ones and research notes were taken in 2007-2008. Notes and transcripts can be found in Appendix C.

Each of them walked through the backend system's development processes involved in the online projects they have undertaken. Participants were assured of the confidentiality of the interview. Online banks observed were HSBC Bank Plc UK, NatWest Bank UK and Barclays Bank Plc (Melomey et al., 2008b). Online games that were observed were puzzle games such noughts and crosses, scrabbles, mazes and war games (Melomey et al., 2007; Melomey et al., 2008b). VLEs observed were UELplus and webct from London university colleges (Melomey et al., 2008c; Durkee *et al.*, 2009). Data collected were analysed for MaMM development.

### **3.4 Simulation**

Simulation has been used in research for many decades to study the use of models and the complex relationships that exist between them. Simulation techniques have been used in many research environments to aid in decision making, to gain more insight into a system, as a guideline for research.

In this research, MATLAB was used as a platform for simulation and testing because it offers a tried and tested scientific and engineering computing software environment. It has been shown to offer a reliable high speed programming environment for a number of computing fields. Since its inception it has been tested widely in several application sites such as fault identification, neural network design, mixed-mode modelling, controller structure selection, parametric and multi-objective optimisation, real-time and adaptive control, parallel genetic algorithms and nonlinear system identification.

### **3.5 Summary**

In this chapter, the various research processes employed in this research are discussed. Results from the Delphi study were used as a strong indication to select complex case studies. The evaluations of these case studies were used to build the Mobile agent-based methodology. Requirements captured from the online cases were autonomy, migration, synchronisation, persistency, concurrency, name services, transparency, fault tolerance and security, however, mobility is the issue for consideration in this research and not security. These sets of requirements run through all the online applications cases. Distributed platform requirements were also discussed which form a base for the development of all online application.

# CHAPTER 4

## Mobile agent Mobility Modelling

### 4.1 Introduction

This chapter presents phases of Mobile Agent-based Methodology (MaMM) to model mobility which provides a platform for analysing, designing and implementing the ever evolving complexities in mobile software and mobile agent based software development. This provides a formal way of presenting mobility concepts and elements in a mobile agent based systems.

### 4.2 The Mobile agent Mobility Methodology (MaMM)

The primary purpose of this methodology is to guide developers through the development of mobile agent-based systems from the point where a business need is identified and approved, to the point of implementation of the system. The way to apply this methodology is indicated in Figure 4.2, where an output from one phase serves as input to the next stage. The methodology is iterative throughout all the phases of systems development. The phases are as follows: Mobility Requirement Elicitation, Fitness Classification, Code Transformation, and Mobility Implementation. This process provides a guide to capturing the business logic and problems underlying complex mobility systems. The research contribution lies in the approach used to address mobility issues in mobile agent-based systems development using fitness functions and a mobility classification model.

### 4.2.1 Reasons for Developing the Mobile agent Mobility

#### Methodology (MaMM)

- Existing methodologies and approaches do not address core mobility issues in systems development such as mobile agent-based applications. Mobility issues such as messaging, location, synchronisation, persistency and migration are poorly addressed or not considered (Belloni and Marcos 2004; Loukil, 2006; Chhetri 2006).
- Current methodologies and approaches such as MaSE, Tropos, GAIA do not address mobility issues in mobile agent systems. These methodologies only focus on multi-agent systems development as discussed in the literature review (Belloni and Marcos 2004; Loukil, 2006; Chhetri, 2006).
- MaMM addresses the dynamic nature of mobility requirements. These mobility requirements are prioritised depending on the application environment, thereby applying fitness criteria in determining the priority scale of mobility which are based on the findings in this research.
- According to the studies conducted using the Delphi study, experts agreed that the MaMM is adaptable in situations where mobility is a key feature and central to the development of an application.
- MaMM has optimisation features embedded in the fitness functions which are applicable to specific application development environments based on expert opinions and judgments.

#### **4.2.2 Mobility Concepts and Design Requirement Considerations**

There have only been a few attempts to model the dynamics of mobility of an agent. According to current literature, the methodologies are inadequate to specifically address and model mobility in the development of mobile agent-based systems. Chhetri et al. (2006) developed an ontology that describe mobility concepts, and the relationships that exist between the concepts to model mobility issues. The concepts did not provide adequate information regarding mobility among different components and/or interactive agents. This is vital to the survival of mobile agents. The core concepts introduced did not provide definitions for agents and mobile agents but suggested that an agent becomes a mobile agent when it is assigned a role, as such mobility is seen as an attribute (Chhetri *et al.*, 2006). Some concepts from multi-agent researchers are relevant to the development of MaMM in this research. This therefore implies that a designer cannot reason about mobility of the agent during the requirement phase of systems development. However, Chhetri et al. (2006) did not specify security of the mobile agent.

The following provides definitions used in modelling mobility in mobile agent-based systems in this research.

## 4.3 Concepts of Multi-agent Systems

### 4.3.1 Existing Concepts of Multi-agent Systems

#### Agents

An agent comprises of code and state information required to undertake computational processes. An agent lives in an environment or platform (White, 1996; Wooldridge and Jennings, 1995). There are two types of agent and they are stationary and mobile agent.

#### Stationary agent

This is an agent that executes code on the same platform that it originated from. This represents the platforms, which provide services and also enforces security (White, 1996; Wooldridge and Jennings, 1995).

#### Mobile agent

A mobile agent is an agent that is able to migrate from one platform to another across a network. It has basic permission(s) that allows it, at the time of creation, to gain access to services that are offered remotely and then sends results back to its home platform. An agent therefore has a creator/owner that keeps a log of its movement history, its resource requirements, its authentication keys, and access permissions (White, 1996; Wooldridge and Jennings, 1995).

#### Platform

A platform provides the basic functions required to program mobile agent. An agent platform therefore provides the computational environment in which an agent operates. A platform will be modelled as networks of computers or nodes, irrespective of size. A platform will be used interchangeably with a node. A platform will offer resource services to other agents that enter it. A platform can be categorised into two types:



1. Home platform

This is the location from where an agent originates (Jansen and Karygiannis, 1999).

2. Host platform

Any platform that a mobile agent can migrate to apart from its home platform (Jansen and Karygiannis, 1999).

### Task

A task is any action or series of actions an agent or mobile agent can perform (Jennings *et al.*, 1998; Zambonelli *et al.*, 2003).

### Goal

A goal is a specific objective an agent aims to accomplish. This is what motivates the agents to meet for a mobility summit, hence establishing a mobility link in order to achieve a goal (Wooldridge *et al.*, 2000; Perini *et al.*, 2002; Chhetri *et al.*, 2006).

### Mobility

There are two types of mobility; weak mobility and strong mobility (White, 1996; Wooldridge and Jennings, 1995; Wooldridge *et al.*, 2000).

1. Weak mobility

In weak mobility, a mobile agent stores no information on previous hosts visited during migration. This type of mobility is suitable for the collection of online data to perform basic control and configuration tasks from the various network elements, which eventually leads to the reduction of network load. Weak mobility copies only the execution code and executes a program from its initial state.

2. Strong Mobility

Strong mobility represents the migration of code, data and state. Strong mobility accumulates and preserves information on all previous hosts visited during its migration. It is also able to process data on any platform while preserving its state and form from previous visits. Strong mobility copies both the code and state and is able to resume execution where it stopped even though it might not have resources on the current platform. Migration ceases when the mobile agent returns to its original starting platform.

Mobility migration therefore depends on the type of application. The number of mobile agents needed per application will also depend on the size of the application. A mobile agent has a goal and to accomplish this means it has to be broken down into tasks. To achieve this goal, mobile agents must have some knowledge, basic permissions to migrate to a platform and access level permission depending on the type of information or assignment. Mobile agents have an itinerary of the migration activities.

#### Permission(s)

Permission(s) will grant the right to execute an instruction or perform an action. This is the ability to create another agent and to grant them rights to use certain resources in a timeframe before termination occurs (Wooldridge *et al.*, 2000).

#### Sleep Mode

This concept affects and monitors changing conditions. This scenario occurs in a situation where an agent puts itself to sleep until such a time that it is triggered by an event. For example, when a mobile agent is dispatched to book a trip for a later date, it goes to sleep until on the day before the flight, it will then wake-up and inform the

traveller on the details of the flight and also communicate other information, such as a delay if there is one (White, 1996; Zhu 2001).

### **4.3.2 Proposed Concepts for Describing Multi-Agents**

#### Resources

Resources are vital if mobile agents are to perform migration tasks on a home or host platform. This includes bandwidth, buffer space, disk storage, network access to file servers and print services and process time.

#### Interactive Events

Interactive events involve establishing a link(s) or a session between or among agents and mobile agents on platform(s). An interactive event occurs if the agents and mobile agents can identify each other regardless of the platform or zone.

For example, an interactive event allows two or more agents and/or mobile agents to meet on the same platform. Here, a mobile agent can decide and migrate to meet another stationary agent on a server platform for a service. Two different agents on a similar mission of booking a flight can meet each other at either the same or different reservation server platform. In this situation mobility summit might be the common place where agents meet for such transactions.

#### Mobility Itinerary

Itinerary represents the mobility plan of the mobile agents' movement.

### Zone

A zone is modelled as a collection or a group of platforms operated by the same authority. A mobile agent should provide enough authorisation and authentication to the destination zone otherwise access will be denied. A mechanism will therefore be provided to verify the authority of a mobile agent migrating from zone to zone. Authority will limit what platforms and agents can do at any point in time.

### Knowledge Base

These are pre loaded information that the mobile agent is dispatched with in order to make a decision. These are migration or route related information.

### Classification Model

This is a mobility determinant model made up of four groups, three of which must be satisfied for an application to be accepted for a mobility application. This is dependent of the requirement elicited for the application.

## **4.4 Composition of Agent System**

Agent system comprises of agents, mobile agents, platforms and resources. Figure 4.1 illustrates the composition of agents, mobile agents, platforms and resources that can form a complete zone. It shows how mobile agents are able to migrate and interact from a home platform to host platform. Each platform is made up of stationary agents, mobile agents and platform resources. Stationary agents can communicate with other stationary agents based on the task assigned to the mobile agents, and also able to gain access to platform resources. Platforms on a cluster, operated by the same authority are referred to as a zone. If a mobile agent migrates to another cluster which is operated by a

different authority then the mobile agent must have that particular zone's authority to have access to the zone's platform resources in order to continue the task assigned to it.

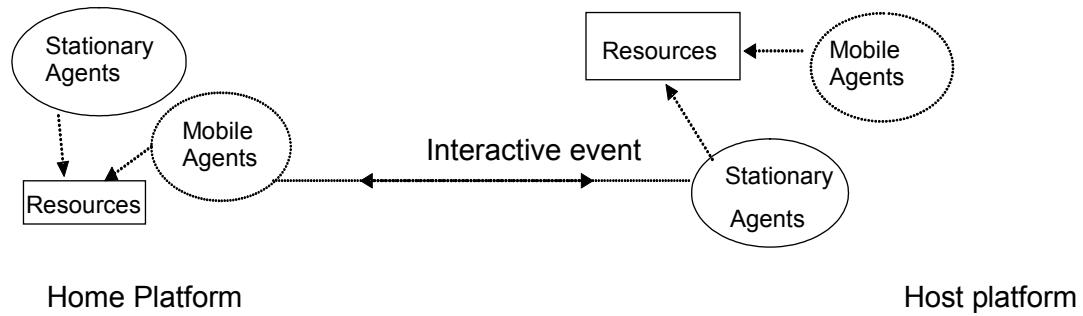


Figure 4.1: Agent-to-Mobile Agent Diagram

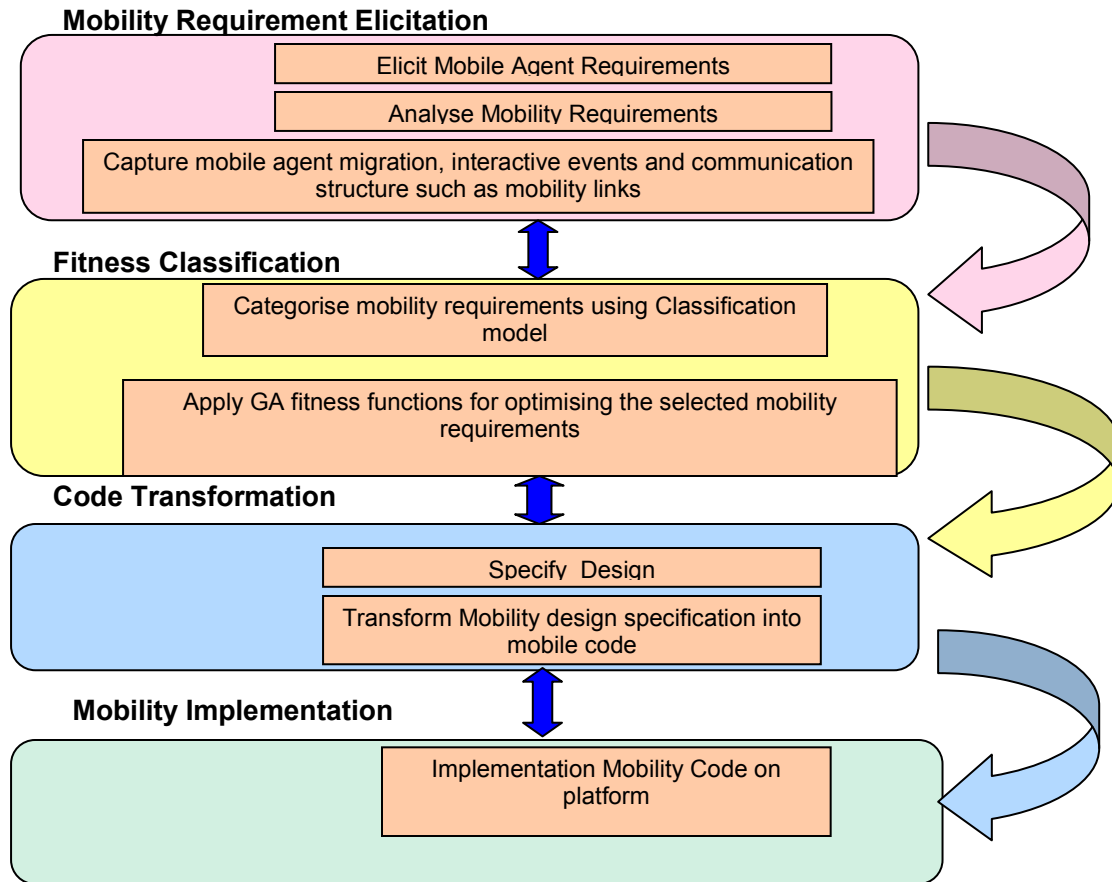


Figure 4.2: Phases of MaMM

## 4.5 Phases of MaMM

### 4.5.1 Mobility Requirement Elicitation

Mobility requirement phases are defined in sufficient detail for example mobility concepts, safety specifications (which include how to use the software), maintenance, data and database definition, security specifications of the mobile agent (which include threats to the mobile agents and platforms), functions, entity types and interfaces are identified for the system design to proceed. This phase includes developing the requirements for the various components of the system and examines and gathers desirable objectives from stakeholders view points. This is achieved to determine *why* an

application is needed, *what* the application will do and *for whom* it is being developed. This phase falls under two categories which are functional and non-functional requirements.

Most systems designers such as real time designers and embedded software designers use IEEE STD 830-1998 as a basis for the majority of the software specification applied to both large and small projects. The standard also provides a baseline for validation and verification. Some of the issues that the standard addresses are functionality, external interfaces, performance and attribute and design constraints imposed on an implementation.

Designers in the mobile agent community also have Object Management Group (OMG) Mobile Agent Facility (MAF) Specification which can be applied to any applications development to make it MAF compliant. MAF is a standard which provides a facility where agents' platforms from different vendors can be interoperable. This facility gives the designer the flexibility of incorporating it with this mobility methodology during the development process.

A systems lifecycle requirement is an iterative process that occurs during the entire process (Sommerville, 2004). This process involves eliciting and analysing the requirement of the application domain. It involves the participation of stakeholders and end users with regard to what is required by the system. In this way, the designer is able to differentiate between the system and user requirements. At this stage and based on information gathered, the designer will be able to cluster the mobility specific

requirement from general requirements of the system. Interactive elements are also identified and documented.

The way to approach the elicitation process is to cluster high interactive events and activities together as this might give an indication of reasoning about mobility. If the proposed system is heavily interactive in combination with other functional elements then building a mobile agent based system is a possible alternative. Reasoning about the mobility and its classification at this requirement phase must be identified before entering into the fitness classification phase of the methodology. A mechanism on how applications and mobile agents could authenticate themselves on the distributed platform should be defined as well as the detection of facilities of a specific network. In reasoning about mobility, another issue worth noting is the classification of business requirements and technology requirements. Under business requirement the developer/designer may consider new business models and remote access to the systems. Technology requirements may also consider the portability of the application under development to suit stakeholder needs, advanced state-of-the-art capability and the distributed architectural environment.

The following guide enables the developer to identify the mobility goals in a given set of system specifications:

- Where client locations are geographically dispersed
- Targeted towards large and highly mobile clients
- Where additional services and service components need to be added in real time
- A need for presence with regard to services that must be available at all time
- High volume of interactive events



- A need for uninterrupted access or reliability in the synchronisation process
- When application is needed for personal productivity in order to achieve real and transformational value, for example in healthcare. This could be from small to large organisational set-ups.

An example of a functional set of requirements for election software (Internet Policy Institute, 2001; Grimm *et al.*, 2006) is;

- Integrity. Casting votes must be correctly tallied. Votes should be easily modified and deleted votes should be detected.
- Auditability and Verifiability. The system should be capable of verifying and tallying all final votes and should demonstrate authentic vote records. The system should also allow for the recount of votes cast.
- Accuracy. There must be multiple backup systems available. Election systems should record votes correctly.
- Transparency. Each voter should have a general knowledge of the voting process
- Eligibility, authentication and uniqueness. One eligible vote per one voter should be allowed to preserve election fairness. A voter should not be able to vote more than once.
- Reliability. The election system should be robust such that there will be no loss of votes counted in the event of power failure.
- Provide an audit trail. The system must contain both paper and computerised backup for recounting votes cast and the total number of votes cast, should a

dispute arise. This means that anonymous records of votes cast will be retained and a record of individual voters, in order to check against ghost names in the electoral register.

#### **4.5.2 Fitness Classification**

This phase deals with analyses of the needs of the user and based on the outcome, the user requirement is developed. A detailed functional requirement and mobility requirement is created which clarifies any discrepancies, conflicts and any misunderstanding that might have occurred during the mobility requirement phase. Models of the mobile agent based systems are produced and refined to reflect the function of the system.

This phase involves the identification of all interactive components and how they link to platform resources and mobile entities. The linkages between mobile agents, stationary agents, resources and platform must be clearly defined. The communication structure to be adopted must also be clearly demarcated. Interaction and behavioral characteristics of mobile agents should also be analysed at this point for adaptability to various devices. This ensures seamless connectivity and transparency in the system. During this phase attention must be paid to data resources that may need continuous synchronisation with the platform. Mobility models developed should specify interaction models, movement capture models and design models that will ensure the systems meet the required specification. At this stage, the intended use of the system is analysed where functional and data requirements are specified. The mobility model indicates how software process and mobile processes interact together in mobile agent-based systems and how these

processes create and use mobile data. When an application to be developed (such an internet voting system or an internet banking application) is found to exhibit functional and non-functional requirements such as autonomy, migration, availability, persistency and messaging, then at least three of the four categories in the classification model must be followed as shown in Figure 4.3. This mobility classification model evolved from the case studies interviews presented in chapter 3.

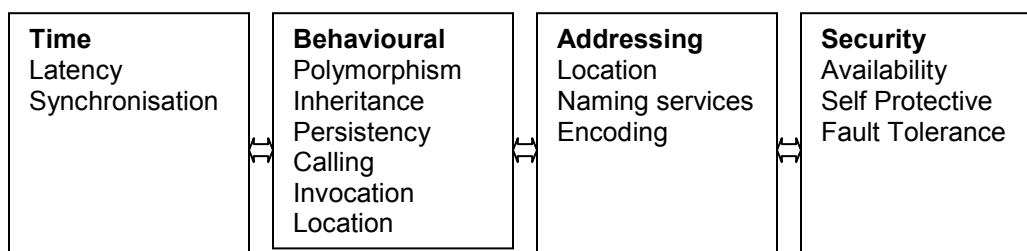


Figure 4.3: Mobility Fitness Classification Model

In this research, mobility platform requirements can be classified into four main categories as indicated in Figure 4.3, these are:

1. Timing requirements - Latency (response times) and Synchronisation
2. Behavioural requirements - Polymorphism, Inheritance, Persistency, Calling, Invocation, location, message passing;
3. Addressing requirements – Location, Naming Services and Encoding,
4. Security requirements - Availability, Self Protective, Fault tolerance and Certified

#### 4.5.2.1 Mathematical Modelling of Mobility Fitness Requirements

##### Addressing

There are certain elements that need to be present for an entity (agent) to be able to travel from its platform of origin  $Hp_i$  to a host platform  $Vp_n$ . These elements are required to perform an address resolution prior to the process migration. The three elements that need to be present are:

- Receiver identification (R.ID)
- Packet identification (P.ID)
- Transmission Frequency of physical layer (TF)

Let  $R$  be the set requirement R.ID, P.ID, TF

Let  $H$  be the set header fields that contains the control information

Let  $L$  be the length of the packet

Let  $p$  be the payload type

Let  $s$  be the sequence numbers

Let  $i$  be the integrity check information

$$R \subseteq H$$

where  $R$  is the set of requirement R.ID, P.ID, TF and  $H$  is the set header of fields that contains the control information

Each computing platform is identified by an assigned address. A process will be able to migrate if it contains a header field carrying the control information. The address resolution client which is the host platform needs to verify the integrity, authenticity and the logical address for resolving information sent across different platforms.

A platform which is hosting each mobile agent needs to ensure that the mobile agents on its platform have a valid server and that the address resolution is also valid. Authorisation of the available address to be used should be provided by both the host platform and mobile agent servers in order to ensure the validity of the address.

### Replication

High availability of services is paramount to mobile distributed computing as this enhances performance. Replication is a technique that is used to maintain copies of data in a geographically dispersed environment and also as a back up in the event of the loss of data or a systems failure (Coulouris, Dollimore & Kindberg, 2005). The fitness of a replica will be measured in real time by the function of the differences in elapsed time. This ensures consistency and correctness at anytime for the events. This is represented as follows:

$$F(t): f_{t+1} + f_{t-1}$$

Where  $f_{t+1}$  is the current time replica server was Created/Accessed/Resolved. This can be expressed as  $f_{t+1} = \{C_{t+1}, A_{t+1}, R_{t+1}\}$

where  $C_{t+1}$  is the current time the replica server was created,

and  $A_{t+1}$  is the current time the replica server was accessed.

and  $R_{t+1}$  is the current time the replica server was resolved.

$f_{t-1}$  is last known time a replica was Created/Accessed/Resolved. This can be expressed as

$$f_{t-1} = \{C_{t-1}, A_{t-1}, R_{t-1}\}$$

where  $C_{t-1}$  is the last known time the replica server was created,

and  $A_{t-1}$  is the last known time the replica server was accessed.

and  $R_{t-1}$  is the last known time the replica server was resolved.

Alternatively, replication can be calculated by;

$$f(x) = \{x_1, \dots, x_n\}$$

where  $x$  is a replica which includes binding relationship variable which are object, location and interface.

Let  $o$  be object/agent

Let  $l$  be location

and let  $i$  be interface

$$f(x) = \{x_{oli}, \dots, x_{oli}\}$$

### Remote Method Invocation

A method is transparently invoked from process A to process B across a network, as if it were a local method, is termed a Remote Method Invocation (RMI) (Coulouris *et al.* 2005; Williams, 2000). This holds true for an object oriented language rather than a procedural language. Invoking a method remotely involves two processes:

1. A reference to the remote object.

2. A registry to store remote references.

Let  $n$  be the number of identified elements for solution  $X$

$x_i$  be elements in  $X$

$f(x)$  the fitness of  $x_i$

The fitness of  $F$  can then be defined as

$$F(X) = \frac{1}{n} \sum_{i=1}^n f(x_i) ; n > 0$$

The average fitness for the elements in the mobility requirements is identified as:

$$F(x): H_{pi} \rightarrow V_{pn}$$

### Persistency

The Object Management Group (OMG) service stipulates a typical structure for persistency. This should consist of persistent Identification (ID), persistent object, persistent object manager, persistent data store and protocol. A persistent object or entity that need to travel from the Home platform  $H_{pi}$  to visit  $n$  number of visiting platform  $V_{pn}$  requires a reference ID, a dynamic state that lives for the duration of the process and a persistent state that will be used for reconstruction of the dynamic state in the case of a failure. These conditions qualify for an entity to be mobile in an environment.

Persistency with respect to transparency needs certain elements to be able to move from one location to another in this case from  $H_{pi}$  to  $V_{pn}$ . These include;

- stability
- recovery
- refining object interface
- activating and deactivation of the object
- relocation

To measure activation and deactivation

Let activation be 1

and

deactivation = 0

then

$$\text{function } F(x) \rightarrow \Delta \frac{d}{a} = \int_{\frac{d}{a}}^{\frac{d}{a} + \Delta \frac{d}{a}} f(x) dx$$

Hence

$$\text{function } F(x) \rightarrow \Delta R = \int_R^{R + \Delta R} f(x) dx$$

### Naming services

The Sun Microsystems naming services system administration guide defines naming services as a central repository that computers, end users, and applications use to communicate together across the network. In this work, we also define name services as integrated services that manage all name information and hierarchies, and also as an autonomous feature for transparency and persistency of entities. The purpose is to provide a basic function and mapping service of name to address on the network. In



order to obtain the remote computer's address, the program must request assistance from say  $H_{pi}$  from the Domain Name Services (DNS) database running on that platform. DNS is a naming service which provides identification for computers on the internet. The name server uses  $H_{pi}$  as part of the request to find the IP address of the remote computer. The name server returns this IP address to the  $H_{pi}$  only if the host name is in its database. It uses a logical tree to resolve names as part of the service

### Synchronisation

Synchronisation is important to maintain consistency of processes from host platform to visiting platform at any given time (Coulouris *et al.*, 2005). The concept of clock synchronisation deals with the understanding of the occurrence ordering of events as produced by the current processes. These events occur between the message sender and message recipient, for example from process A to process B. Clock synchronisation is required to provide the mechanism that can assign numbers sequentially based on the agreement between the sending and receiving processes. Several algorithms have been developed to achieve this over past decades. Lamport (1978) introduced the concept of an event happening before another in distributed environment. The notion is illustrated between event  $a$  and  $b$ ;  $a \rightarrow b$  where event  $a$  'happens before' event  $b$ . Another algorithm developed by Lamport and Meilliar-Smith (1985) requires a reliable connected network to handle a fault situation. Christian (1989) developed an algorithm which measures the local time at which a message is sent  $T_o$  and the time at which a message is received  $T_1$ . This is done by issuing a remote procedure call to a time server to obtain the time. The delay in the network is then estimated as  $\frac{T_1 + T_o}{2}$ . Hence

the new time can be said to be the time returned by the server in the addition to time elapsed by the server to generate the timestamp. This is expressed by

$$\text{Time new} = \text{Timeserver} + \frac{T_1 + T_0}{2}.$$

The Berkeley algorithm which was developed by Gusella and Zatti (1989), was based on the assumption that any computer on the network has an accurate time which can be used for synchronising time between processes. This assumption can introduce delays and losses depending on the network which is due to the distributed nature in accessing the network and the processing capabilities on the learning system.

Let S be Synchronisation

Hp be Home platform

Vp be Visiting platform

Vp<sub>n</sub> be n visiting platform

P<sub>n</sub> be n number of processes

The timescale for measuring change in synchronisation is  $\delta s$  important where s

(Synchronisation) is a derivative of the  $f(x)$  which is  $\frac{\delta f}{\delta s}$ . Measuring the short time for  $n$  process is dependent on how fast changes occur in the system. The time range between which  $n$  process leaves Hp and arrives at Vpn can be expressed as:

$$F(x) \rightarrow \delta t = \int_t^{t+\delta t} f(s) dt \text{ where the interval is } [t, t + \delta t]$$

such that if  $f$  is a continuous real value function defined by the limits  $[t, t + \delta t]$  and

hence

$$F(x) \rightarrow \Delta t = \int_t^{t+\delta t} f(s) dt = F(t + \delta t) - F(t)$$

where  $F(x)$  is a complex system during its evolution

Function synchronisation is dependent on [time, location]

This produces elements for the mobility fitness to be selected for the mobile agent-based system development and the associated documentation. An interaction analysis is performed to define series of interaction between business activities and data. Before the mobility design is achieved detailed analysis and a model regarding the user and systems interaction must be produced, the platform location and constraints must be specified, messaging must also be clearly defined.

This expression therefore represents a fitness function in an inverse relationship to a fitness solution.

Let  $F(X) = \{x_1, \dots, x_n\}$

The fitness function  $U(m) = \{x_1, \dots, x_n\}$

Where  $U(m) = (1/e+x)^2$

A designer can also employ combinatorics to choose a set of mobility elements from a large set of distributed systems requirements for complex systems which is known as choose function (n choose k).

#### 4.5.2.2 Binomial Coefficient Application to Requirements

Table 4.1 lists both generic and mobile requirements for the development of distributed system. The list is not only limited to what the table provides so a developer has the choice of adding more requirement elements to the list depending on the type of application. The requirements for developing mobile distributed systems were derived and assigned bit strings in order to be able to apply binomial coefficient to provide

solution. Binomial coefficients  $\binom{n}{k}$  also known as choice number or choose function are

read as ' $n$  choose  $k$ ' (Conway and Guy, 1996). Combinatorics is a branch of mathematics which is concerned with solving problems and in computer science it is used for estimating the number of elements of certain sets. In the case of ' $n$  choose  $k$ ', this is interpreted as the number of ways of picking  $k$  from the unordered outcomes of  $n$  number of possibilities.

Generic & Mobility Distributed Requirement	Variable Number	Bit Strings	Mobility Fitness Function Representation Element(F)	Binary Value
Abstraction	1	1	f3	10000001
Addressing	2	1	f11	10000010
Availability	3	1	f13	10000110
Calling	4	1	f7	10001100
Concurrency	5	0	n/a	10001110
Encoding	6	1	f8	10011110
Fault Tolerance	7	0	n/a	10000111
Inheritance	8	1	f5	10000011
Invocation	9	1	f8	11000011
Latency	10	1	f2	10011001
Location	11	0	n/a	10100011
Message Passing	12	1	f9	10110001
Mobility	13	0	n/a	10001011
Naming	14	1	f10	11010011
Openness	15	0	n/a	10010010
Persistency	16	1	f6	11011011
Polymorphism	17	1	f4	10101001
Replication	18	1	f14	10011011
Resource Sharing(Scheduling)	19	0	n/a	10000100
Scalability	20	0	n/a	11111000
Security	21	0	n/a	11001100
Self Protective and Certified	22	1	f15	10011010
Synchronisation	23	1	f1	11100010
Transparency	24	0	f16	10111000

Table 4.1: Generic and mobility requirements

For example  $\binom{4}{2}$  gives 6 as the number of possible combinations of two elements that could be derived from the set of numbers  $\{1,2,3,4\}$ . This will be  $\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}$ . These six combinations are known (in binomial) as  $k$ -element subsets of an  $n$ -element set; hence this is the number of ways  $k$  combinations can be taken from a set of  $n$  elements. The binomial coefficient is therefore implemented as  $\text{binomial}[n, k]$ . The value of the binomial is usually represented by

$${}_nC_k \equiv \binom{n}{k} \equiv \frac{n!}{k!(n-k)!} = \frac{n(n-1)(n-2)\dots(n-k+1)}{n!}$$

For example, the requirement

element in table 4.1 can be represented by real numbers with mobility elements represented as even numbers in the table. In translating this in to MATLAB;

let C be all combinations

C = nchoosek (n,k) where n and k are non-integers which returns the value  $\frac{n!}{k!(n-k)!}$

C= nchoosek (v,k) where v is the row vector with length n which creates a matrix of all possible combinations of all n requirement elements . The matrix C will contain  $\frac{n!}{k!(n-k)!}$

rows and k columns. From the MATLAB command windows, for every two sets of requirement elements one of them is a mobility requirement element, then the command nchoosek (2:2:24,12) returns the even numbers from two to twenty-four, taken twelve at a time:

```
>> nchoosek(2:2:24,12)
```

```
ans =
```

```
Columns 1 through 10
```

```
2    4    6    8   10   12   14   16   18   20
```

```
Columns 11 through 12
```

```
22   24
```

From table 4.1 nchoosek (2:2:24, 12) translates to the following requirements in order in which they occur; addressing, calling, encoding, inheritance, latency, message passing, naming, persistency, replication, scalability, self protection and certified, transparency.

Alternatively if we choose a small data set then the command `nchoose k (2:2:12,3)` returns the even numbers from two to twelve, taken five at a time

```
>> nchoosek(2:2:12,5)
```

```
ans =
```

```

2   4   6   8  10
2   4   6   8  12
2   4   6  10  12
2   4   8  10  12
2   6   8  10  12
4   6   8  10  12
```

These combinations represent different combinations of mobility requirements highlighted in table 4.1. From table 4.1 choosing a small data set with the command `nchoosek (2:2:12,3)` translates to:

Addressing	calling	encoding	inheritance	latency
Addressing	calling	encoding	inheritance	message passing
Addressing	calling	encoding	latency	message passing
Addressing	calling	inheritance	latency	message passing
Calling	encoding	inheritance	latency	message passing
Calling	encoding	inheritance	latency	message passing

Whenever a binomial coefficient is expressed as a gamma function such as  $z! = \Gamma(z + 1)$ , binomial coefficients are generalised in a way that allows non-integers to be expressed as arguments which includes complex numbers for  $n$  and  $k$ . Gamma is implemented as  $\Gamma(z)$  and gamma function is a natural extension of factorial to all complex and real number arguments (Arfken, 1985).

To translate for use into MATLAB, the identified generic and mobility requirements in Table 4.1 will be represented as a set of  $n$  elements, while the mobility fitness element is represented as  $k$  combinations of a set of  $n$  elements.

The next section explains how the concept of GA is used in formulating the mobility problem and evaluating fitness criteria using fitness functions modelled in section 4.5.2.1.

#### **4.5.2.3 Concepts underlying GA Problem Formulation:**

##### Genetic Algorithms (GA)

GA is a search method motivated by evolutionary biology where evolution models are formed based on crossover, mutation and a selection process (Goldberg and Deb, 1991). This random search method provides effective solutions to optimisation problems in computing. The solutions are usually represented in a binary bit string.

Historically, GAs can be linked to Holland (1975) who described the ability to encode complex structures into a bit strings to make it more manageable. He also explained that



with appropriate control structures, rapid improvement can be made under transformation conditions such that a population of bit string can evolve as in a similar way of the animal population (Bonner *et al.*, 1996). GAs have been applied to several application problems as an aid in finding the optimal solutions to problems, thereby reducing costs and maximising efficiency in many industrial applications.

The evolution begins with a randomly generated population of individuals which usually happens in generations. The fitness of every single individual in the population is evaluated and multiple individuals are randomly selected based on their fitness from the current population and are modified to form a new population (Fogel, 1995). This new population of individuals is used in the next iteration of the algorithm. The algorithm is normally terminated when the maximum number of generations has been reached. GA has two main components which are the genetic representation of the domain solution and the fitness function used to evaluate the solution (Fogel, 1995).

The following are important terminologies associated with GA and these are population, chromosomes, genes, genotype, phenotype and candidate solution:

### Population

A population is an abstract representation called chromosomes which is also known as individuals is used to optimise problems which can evolve into better solutions (Back, 1996).

### Chromosome

A chromosome is made of a very long strand of deoxyribonucleic acid (DNA) and contains many genes of about hundreds to thousands. Genes consist of DNA which

contains the code used to synthesize a protein. Genes vary in size, depending on the sizes of the proteins for which they code. The genes on each chromosome are arranged in a particular sequence, and each gene has a particular location on the chromosome. In addition to DNA, chromosomes contain other chemical components that influence gene function (Holland, 1975).

### Genotype

Genotype is the gene type of an organism; the alleles of a certain characteristic. The genotype is the genetic makeup, for example all of your genes are what comprise your genotype. The genotype is a person's unique combination of genes or genetic makeup. Thus, the genotype is a complete set of instructions on how that body is supposed to function and be built (Holland, 1975).

### Phenotype

Phenotype is the way genes express themselves example short, tall, or green. The expression of genes is called phenotype, the traits that results when the instructions in your genes are carried out or expressed. The phenotype differs to some extent from the genotype because not all the instructions in the genotype may be carried out or expressed. Therefore how a gene is expressed is determined not only by the genotype, but also by the environment which includes illnesses and diet and other factors (Goldberg, 1989).

### Candidate solution

Candidate solution is the possible solution usually represented with a bit string. An example of a set of candidate solution is

100011101001111010011001111000101000101110011011

The above represents possible solutions within which a solution could be found.

### Genetic Operators

GAs has two important operators which are crossover and mutation (Baker, 1985; Goldberg and Deb, 1991; Haupt and Haupt, 1998; Larranaga *et al.*, 1998; Wilson *et al.*, 2003 and Ware *et al.*, 2003). To prepare data for GA, data must be encoded and encoding a chromosome should in a way represent information about the solution. Encoding is represented in a binary string.

### Crossover

Crossover is also known as recombine. After deciding on the encoding to use, the genetic operator selects genes from parent chromosomes to create a new offspring or a child. The easiest way to achieve this is to select a crossover point randomly (Ray and Bandyopadhyay, 2005; Tsai *et al.*, 2002) then everything before this point is copied from the first parent and everything after the crossover, from the second parent, is also copied to form a new child or offspring. More decisions regarding the crossover can be made based on the following; the complexity of the problem, the encoding of the chromosome and the level of experience of the designer.

### Mutation

Mutation takes place after a crossover has been performed. Mutation changes offspring randomly (Goldberg, 1989; Lima *et al.*, 2005). With binary encoding, selected bits could be randomly switched from 0s to 1s or 1s to 0s.

GA is a model based on natural selection and evolution where the stronger individual in the population survives into the next generation. The GA tool which utilises GA principles have a number of benefits which have made it a popular choice for many applications particularly in engineering and employs easy to understand techniques for providing solutions to complex problems. For example, GA and the Direct Search (GAD) tool box provide a platform where functions can be defined with several variables. GAD tool box also have constraint handling capabilities which are encountered in the course of formulating solutions to optimisation problems and can be better handled than the traditional mathematics optimisation techniques.

In this research, population, chromosome, genes, phenotype, genotype and candidate solution are represented using bit string from Table 4.1.

### Problem Formulation

In this research, in order to enable mobility on an agent platform, the core requirements for modelling mobility are represented and modelled using the principles of Genetic Algorithm. Mobile agent applications are usually built and deployed on distributed platforms. There are three important requirements which are critical and essential to mobility on such a platform. These requirements are remote method invocation, synchronisation and persistency. Current literature on approaches and methodologies for modelling agent systems has failed to capture these three essential requirements that enable mobility in mobile agent systems. Hence, this methodology developed in this research offers industry practitioners the opportunity of developing a mobile agent based system that captures persistency, invocation and synchronisation as essential mobility requirements for modelling related applications.

This section is the mobility problem formulation using GA concepts briefly explained in section 1. The initial population of the distributed system requirements is represented by the following chromosomes set;

```
1000000011000001010000110100011001000111010011110100000011000001111000
0111001100110100011100110001100010111101001110010010110000111010100110
011011100001001111100011001100100110101110001010111000
```

#### **4.5.2.4 Representation of Chromosomes in the Mobility Problem Formulation**

##### Chromosome

For example a chromosome is denoted by the bit strings as indicated in Table 4.1;

=100011101001111010011001111000101000101110011011

The above represents possible solutions within which a mobility solution could be found.

##### Genotype

Genotypes representations are bit string encodings for all the candidate solutions. From Table 4.1 each requirement has an associated binary value which is used for encoding the candidate solutions for example;

Addressing and availability, then invocation and calling and location

= 1000001010000110110000111000110010100011

Concurrency and encoding, latency and synchronisation, and mobility and replication

=100011101001111010011001111000101000101110011011

. The search space is  $-0.5 \leq x \leq 0.5$  and the fittest will always be within the boundaries of the range. This is indicated by the final point co-ordinate at which termination occur as tabulated in Table 5.1.

### Phenotype

Phenotype representations are the combination of one or more of the candidate solutions. Each of the genotype is assigned a fitness to evaluate its accuracy, for example in Table 4.1 the mobility requirement is each assign f1 to f16 which has modelled. The fitness function is a derivative of the criteria for specifying fitness. The following functions have been derived using the general GA fitness function to evaluate the genotype to create new individuals;

$$F(X) = \{x_1, \dots, x_n\}$$

The fitness function  $U(m) = \{x_1, \dots, x_n\}$

$$\text{Where } U(m) = (1/e+x)^2$$

At the point of termination each requirement/variable is assign a fitness score known as final point co-ordinate. The best individual(s) falls within the range  $-0.5 \leq x \leq 0.5$ .

1. F1 is fitness representation for candidate solution of synchronisation. The fitness function for evaluating this candidate solution is as follows:

$$F(x) \rightarrow \delta t = \int_t^{t+\delta t} f(s) dt \text{ where the interval is } [t, t + \delta t]$$

such that if  $f$  is a continuous real value function defined by the limits  $[t, t + \delta t]$

2. F8 is fitness representation for candidate solution of invocation

$$F(X) = \frac{1}{n} \sum_{i=1}^n f(x_i) ; n > 0$$

3. F6 is fitness representation for candidate solution of persistency

$$F(x) \rightarrow \Delta \frac{d}{a} = \int_{\frac{d}{a}}^{\frac{d}{a} + \Delta \frac{d}{a}} f(x) dx$$

$$\text{and } F(x) \rightarrow \Delta R = \int_R^{R + \Delta R} f(x) dx$$

### Candidate solutions

Candidate solutions are the individuals' solutions in the genotype. These individuals are possible solutions to the mobility problem. An example of a large set of candidate solution from Table 4.1 is

```
1000000011000001010000110100011001000111010011110100000011000001111000
0111001100110100011100110001100010111101001110010010110000111010100110
011011100001001111100011001100100110101110001010111000
```

The above represents possible solutions within which a mobility solution could be found.

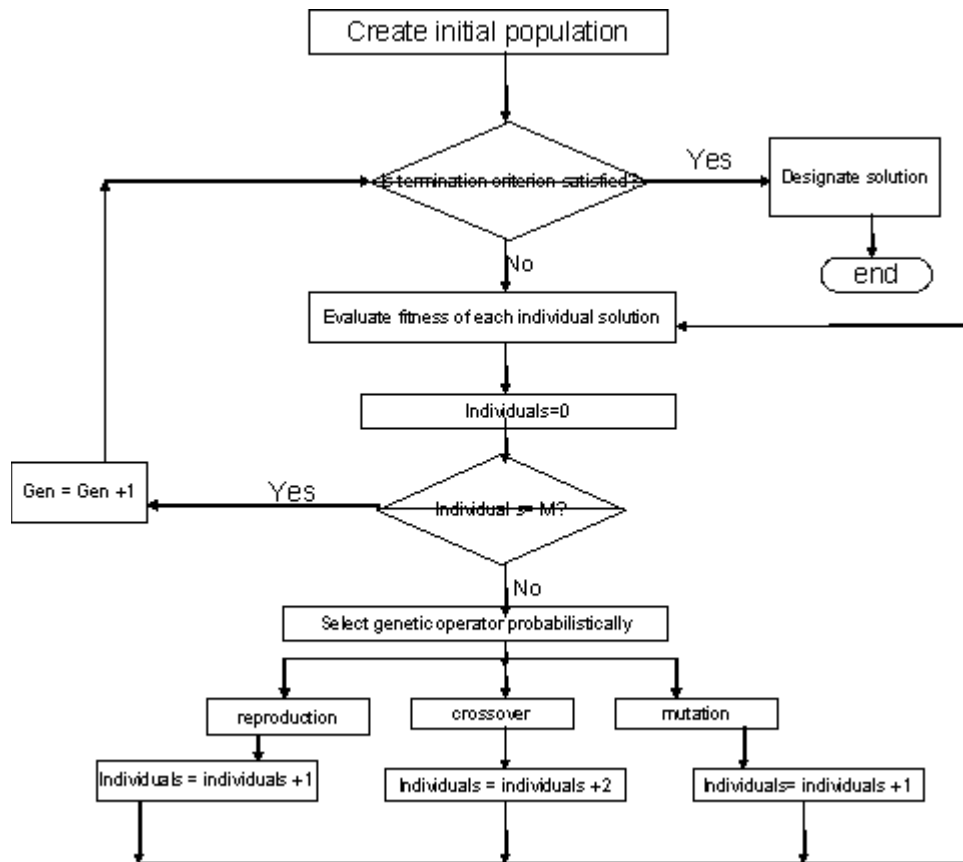


Figure 4.4: Genetic Algorithm Flowchart

The steps towards the solution of the mobility problem are formulated in the following and also in Figure 4.4:

- 1) Generate an initial population of random individuals, made up of variables in Table 4.1
- 2) Perform the following sub-steps iteratively until the maximum number of generation is reached, or the termination criterion has been satisfied:
  - a) Using the following mobility fitness function, each at a time to evaluate each candidate solution:



- '@mobilitysync',
- '@mobilityRMI' and
- '@rastriginsfcn'.

Rastrigins function '@rastriginsfcn' is being used for benchmarking because it is often used for testing Genetic Algorithms. The function point of Rastrigin computes and generate a given number of different points inside the function domain performing searches on each variable or genes and retaining the best results in this case the fittest individuals. It is used for optimising solutions in Genetic Algorithm due to the fact that it performs well with a high number of variables with high reliability.

b) Create a new population by applying the following genetic operators:

- Reproduction: a randomly chosen individual is copied from the current generation to the next.
- Crossover: operates on two individuals in the population, and produces two new offspring
- Mutation: Create a new offspring by mutating a chromosomes.

3) If the termination criterion is satisfied, or the maximum number of generations is reached, the current best individual in the population is proposed as the mobility solution to the problem.

## PSEUDO CODE

Algorithm GA is

```
// start with an initial time

t := 0;

// initialise a random population of individuals
initpopulation P (t);

// evaluate fitness of all individuals in population
evaluate P (t);

// test for termination criterion (fitness for mobility)

while....
    do.....

// increase the time counter
t := t + 1;

// select sub-population for offspring production
P' := selectparents P (t);

// recombine the "genes" of selected parents
recombine P' (t);

// flip the mated population stochastically
mutate P' (t);
```

```

        // evaluate it's new fitness
        evaluate P' (t);

        // select the survivors from actual fitness
        P := survive P,P' (t);
    od
end EA.

```

### 4.5.3 Code Transformation

This phase establishes the physical characteristics of the operating environment. Major subsystems and their mobility inputs and outputs are defined and migration processes are allocated to resources. This phase should also provide a mechanism on how the core mobility element will interface between various applications and other resources on the distributed platform environment. A typical layer diagram guide for the mobility designer is shown in Figure 4.5. The diagram is made up of distributed platform layers which provide the base environment for all mobility application developments such as the mobile agent. The top layer is the mobility platform layer for the mobile agent functionalities and above the top layer is the internet /online applications as illustrated in Figure 4.5.

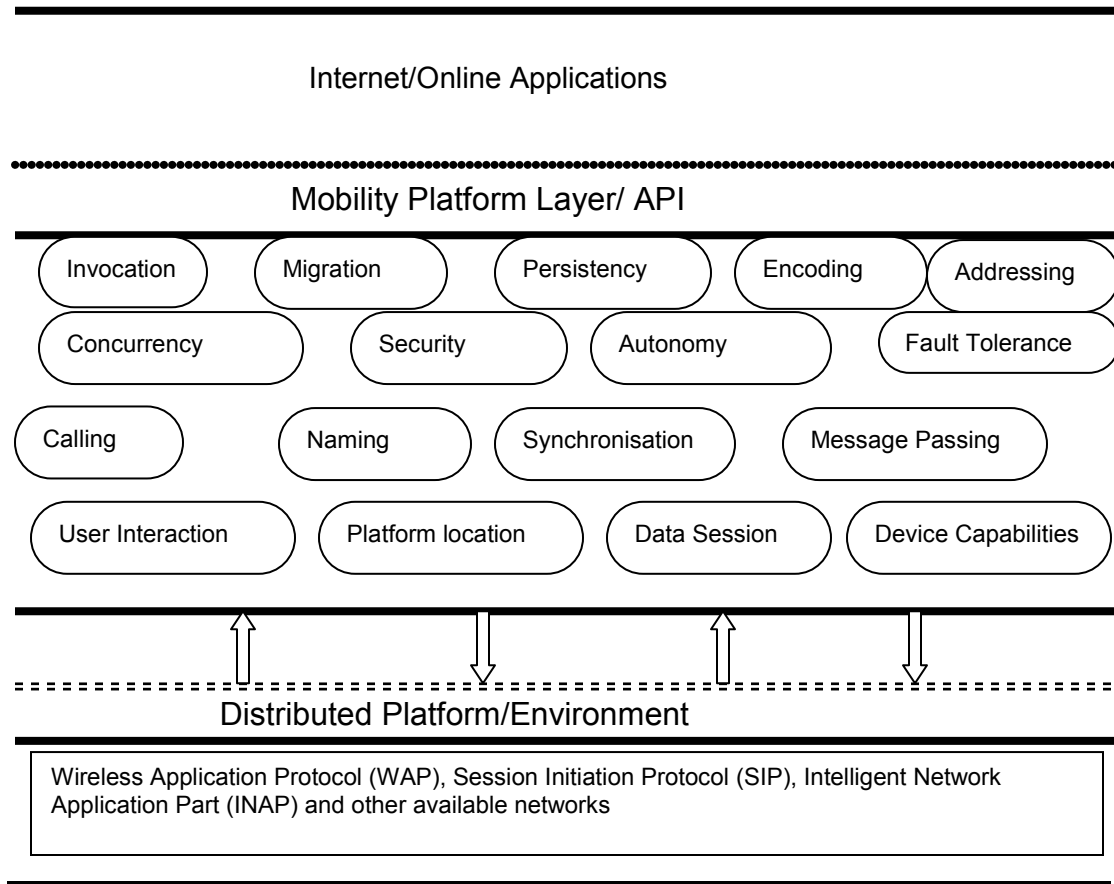


Figure 4.5: Mobility design layer diagram

In this phase the designer has the option of integrating the favoured development environment such as proprietary or standard J2EE or mobile agent development tool. For example a library and a preferred application framework could be added such as Java, C++, Eclipse and others. Various categories of testing can also be developed at this point. For example simulating the environment and testing for interoperability of the system.

#### **4.5.4 Mobility Implementation**

Mobile code is a an executable code which runs in remote locations and require some form of security check before the execution takes place on the platform. The mobile code must be authenticated and authorised by the platform where it intends to execute which is as a results of the complexity of interactions between mobile agent components. Security issues such as threat to the mobile agent and trust related issues will be developed as future work for this mobility implementation as the focus of this research is on mobility.

#### **4.6 Simulation (How the GA tool works)**

The evaluation of mobile agent based system is very challenging and hence requires a carefully planned methodical approach and the selection of a suitable tool to accomplish the selection of the mobility requirements. When a suitable methodology or approach has been identified, the next step is to evaluate it by using simulation or prototyping or alternatively a combination of these. These two evaluation approaches both have their advantages and disadvantages for the mobile agent-based system. Evaluation using a prototyping method has the advantage of demonstrating the feasibility of a proposed system. Prototyping demonstrate how a system will work in the real environment and provides the opportunity to improve the current functionality of the system, which can be discarded when the actual system is built. This provides a limited approach due to the fact that prototyping may not necessarily be translated to a large scale situation and may not function satisfactorily in a real situation. Evaluation of the mobile agent based system using simulation, enables the assessment and measurement of the systems functionality, performance, robustness, scalability, validity and many other measurable features of a system. However, simulation on its own is not able to capture all the vital

aspects of the system under consideration. Simulation models more often than not are simplified version of the actual system, which means that some of the important features and functionality may have been omitted.

## **4.7 Simulation of Fitness Function Using GA Concepts**

### **4.7.1 Objectives of Simulation**

The aim of the simulation is to test the fitness of each mobility requirement variable in a population using the Genetic Algorithm (GA) concepts. The fitness functions evaluated were synchronisation, invocation and persistency. The aim of the evaluation was to assess the performances in each function with respect to best individuals in a given population.

The criteria for assessment were;

- best individuals in relation to best and mean fitness
- objective function value in relation to the number of iterations
- objective function value with approximation and boundaries between 0 and 1

### **4.7.2 Overview of Simulation**

This simulation uses principles of Genetic Algorithms (GA) for optimising mobility requirements. This GA mimics the principle of biological evolution which can modify a population of individuals using genetic operators such as selection, mutation and crossover as explain 4.5.2.1.

Figure 4.6 indicate all features present in the Genetic Algorithm and Direct Search (GADS) Toolbox 2.4.1 that are use to simulate optimisation problems. The GADS Toolbox provides standard algorithm options for solving complex problems and is accessible through both Graphical User Interface (GUI) and the MATLAB command line window. The GUI enables the user to define a problem, select algorithm option and monitor progress and performance. In this research, GUI is used and the algorithm option was substituted with user-defined mobility fitness function. The progress and performance of variables were then monitored. The GA optimization tool provides the options of creating population, choosing and applying genetic operators such as parent selection, crossover and mutation.

Algorithms can be customised by providing user-defined functions. Problems can also be represented in a variety of formats including variables that are a mixture of integer and complex numbers. In this research, user-defined functions thus the mobility fitness functions were used. Fitness functions can also be vectorised in some cases to improve the execution speed. There are also features to allow for the automatic code generation of the optimised solution in the m-file. The automatic code can be exported and run from the command line, if required, to preserve the work or to generate routines.

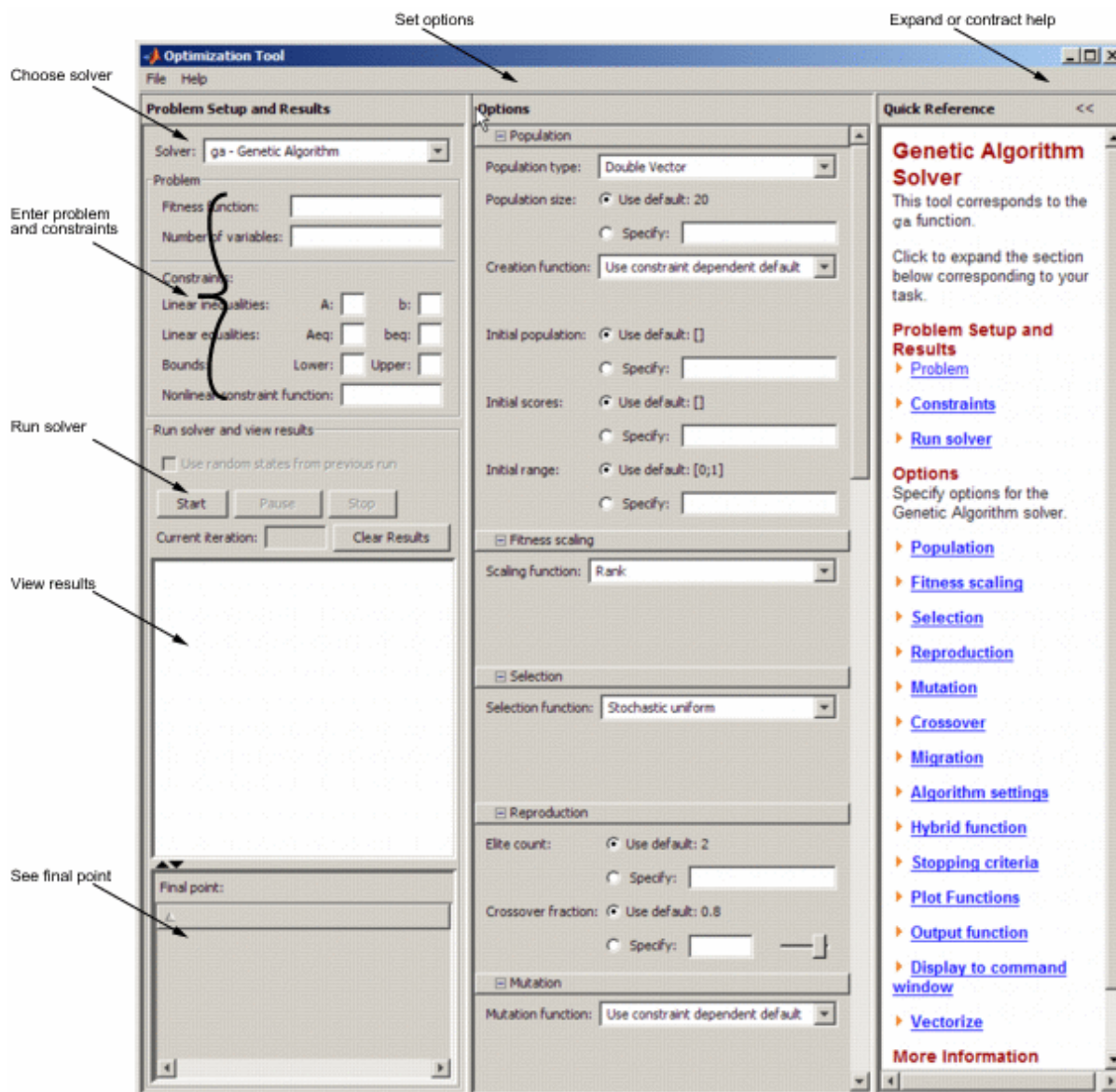


Figure 4.6 GA Optimization tool

MathWorks, <http://www.mathworks.co.uk/help/toolbox/gads/f6453.html> (Accessed 27/01/12)

## 4.8 Summary

In this chapter, the methodology for modelling mobile agent-based systems has been presented. Functions for the requirement elements were formulated and discussed. The MaMM showed how Genetic Algorithm principles can be used to select mobility



requirements for application development as presented in the various case studies. The GA tool which utilises GA principles demonstrates how the required variables can be selected, crossed over and mutated in order to achieve system goals. Given this background insight, chapter 5 will present the testing and simulation results of the MaMM.

# CHAPTER 5

## Simulation Testing and Evaluation

### 5.1 Introduction

The GA tool which utilises GA principles was used to simulate the mobility variables selection in a given population using two solvers which were modelled mathematically using mobility requirement parameters. These mathematical models were further translated into Matlab fitness functions which were @mobilityRMI and @mobilitysync while Rastrigins' function a well known Genetic Algorithms solver for measuring performance (Rastrigin and Erencheyn, 1975), was used to benchmark the mobility function solvers. The results are analysed and discussed in section 5.6.

### 5.2 Mobility Fitness Functions Testing

The simulation process involves the following steps:

1. The first step involves selecting the GA from a list of 'solvers' and specifying the 'fitness function', 'constraints' (if there are any) and the 'number of variables'.
2. The second is to decide which 'options' are appropriate for the simulation. The options include 'population', 'fitness scaling', 'selection', 'reproduction', 'mutation', 'crossover' and 'plot functions'.
3. The third step is to start the simulation while observing the results.

### 5.2.1 Step 1: Problem Setup

In this research, the 'solver' is the GA and the 'fitness function' will be '@mobilityRMI', '@mobilitysync' and '@rastriginsfcn' while the 'number of variables' is set at 20 which represents the number of mobility requirements. These are represented as shown in Figures 5.1, 5.2, 5.3 and 5.4 respectively.

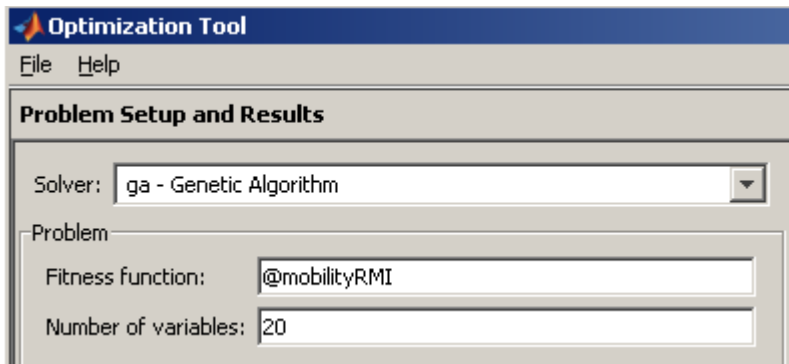


Figure 5.1: MobilityRMI function simulation setup

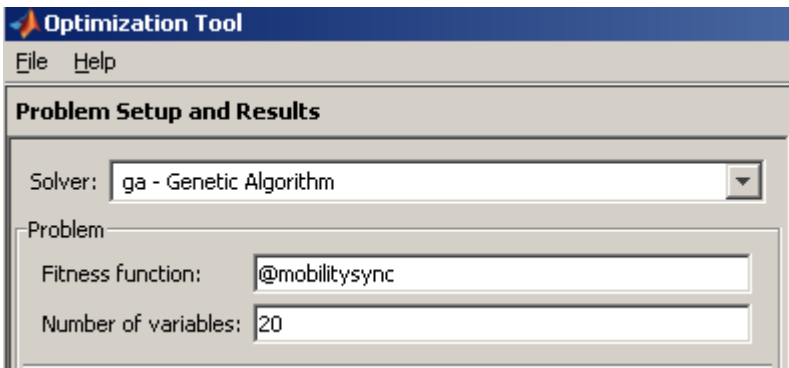


Figure.5.2: Mobilitysync problem setup

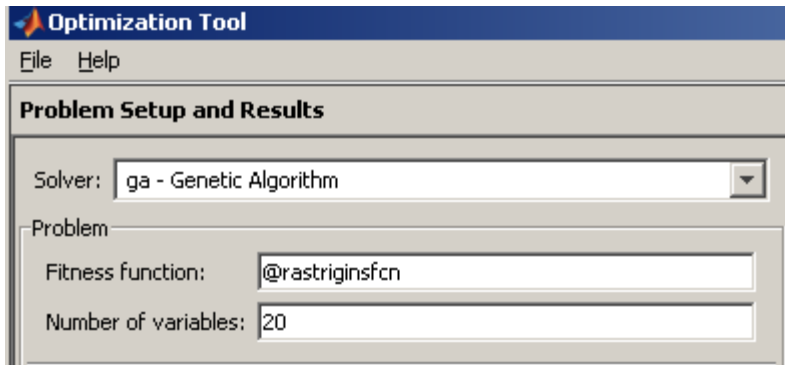


Figure 5.3: Rastrigin's problem setup

### 5.2.2 Step 2: Function Options for Problem Setup

The next step is to specify the type and size of population as indicated in Figure 5.4.

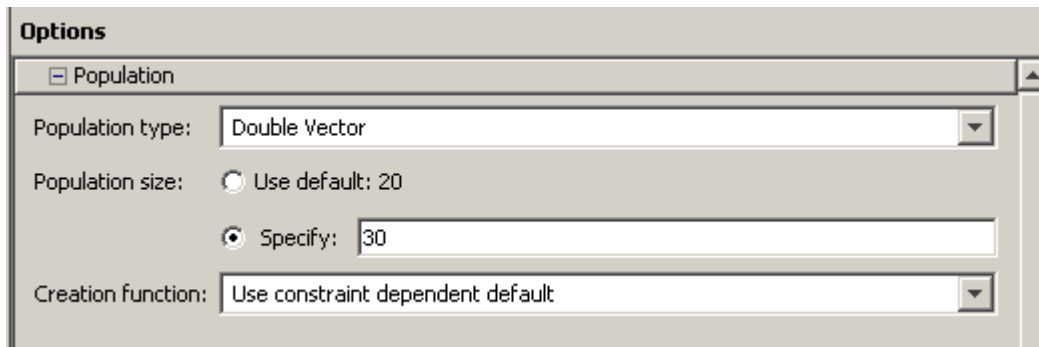
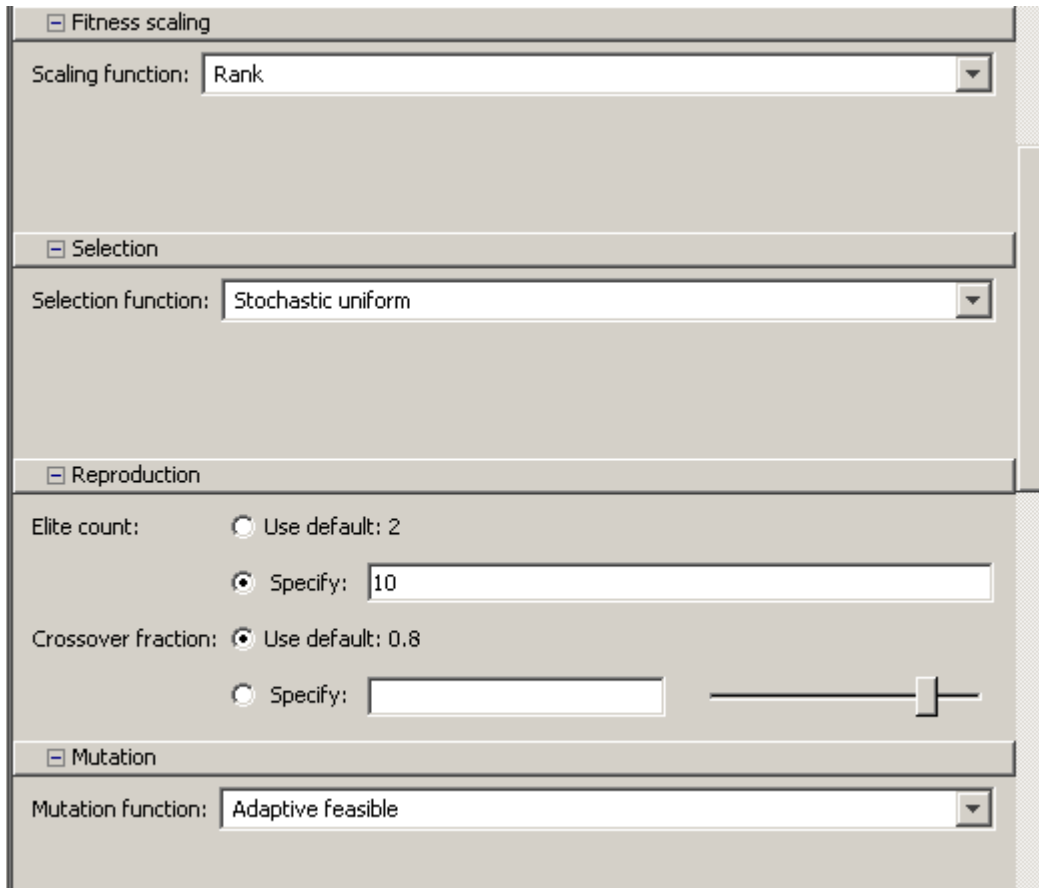


Figure.5.4: Population options

Figure 5.5 indicate the options that are selected and these are 'ranking', 'stochastic uniform', 'elite count' and 'adaptive feasible'. 'Rank' is a 'raw fitness score' that is graded according to the position of each individual. 'Adaptive feasible' ensures adaptability in terms of both the successful and unsuccessful generation.



The image shows a software interface for configuring genetic algorithm operators. It is divided into four main sections, each with a title bar and a collapse icon:

- Fitness scaling:** Contains a dropdown menu for 'Scaling function' set to 'Rank'.
- Selection:** Contains a dropdown menu for 'Selection function' set to 'Stochastic uniform'.
- Reproduction:** Contains settings for 'Elite count' and 'Crossover fraction'.
  - 'Elite count' has two radio buttons: 'Use default: 2' (unselected) and 'Specify: 10' (selected).
  - 'Crossover fraction' has two radio buttons: 'Use default: 0.8' (selected) and 'Specify: [empty box]' (unselected). A slider control is positioned to the right of the 'Specify' box.
- Mutation:** Contains a dropdown menu for 'Mutation function' set to 'Adaptive feasible'.

Figure 5.5: GA Tool GUI for genetic operator options

### 5.2.3 Step 3: Monitoring and Observation

The options made for the 'plot functions' are the 'best fitness', 'best individual' and 'stopping criteria'. These tests indicate the optimised visual results for the fitness of each individual while the 'solver' is still running. The purpose of the 'plot function' is to plot the various aspects of the Genetic Algorithm during execution. The 'best fitness' plots the 'best fitness value' in each generation against the 'number of iterations', while the 'best individual' plots all vector entries of each individual with the 'best fitness function value'. The 'stopping criteria' plot the 'stopping criteria levels'. After the simulation has been set-up and the options defined, the 'solver' is run and results are shown in the view results window together with the number of iterations for observation and monitoring.

### 5.3 Fitness Function Evaluation

In this section, the mobility is modelled mathematically as part of the core mobility requirements which were identified as remote method invocation, synchronisation and persistency. These were translated into mobility fitness functions similar to the Rastrigin function. Rastrigin's function was used for benchmarking performance of the mobility fitness function. Rastrigin's function is a widely accepted function for testing performance of Genetic Algorithms and the search space is  $-5.12 \leq x \leq 5.12$ , however, in this research the search space is scaled down to  $-1.0 \leq x \leq 1.0$  to allow for some margin of error for comparison in the selection of mobility variables. The variables for persistency were integrated in the remote method invocation in order to complete the fitness function component of the remote method invocation. The following fitness sections 5.3.1 to 5.3.3 indicate a step-by-step translation of mathematical models for remote method invocation, persistency and synchronisation into mobility fitness functions which were used for the simulation:

#### 5.3.1 Test 1: Mobility Remote Method Invocation

##### MOBILITY REMOTE METHOD INVOCATION FUNCTION

The following mobility fitness function of Remote Method Invocation (RMI) defines a function:

where  $f(x)$  is the fitness of  $x_i$

The fitness of F can then be defined as

$$F(X) = \frac{1}{n} \sum_{i=1}^n f(x_i) ; n > 0$$

This fitness of F was translated in the following fitness function solver for use with the GA tool with the following lines of codes;

```
function scores = mobilityRMI(pop)

%Author: Divina Melomey
%University of East London
% mobilityRMI compute mobility RMI function
%22/06/2009

scores = 1/30.0 * size(pop,2) + sum(pop.^2);
```

The simulation was defined as @mobilityRMI fitness function as shown in Figures 5.1 to 5.5.

### 5.3.2 Test 2: Mobility Synchronisation Function

#### MOBILITY SYNCHRONISATION FUNCTION

The function F(x) was expressed as:

$$F(x) \rightarrow \delta t = \int_t^{t+\delta t} f(s) dt \text{ where the interval is } [t, t + \delta t]$$

such that if  $f$  is a continuous real value function defined by the limits  $[t, t + \delta t]$ .

Hence

$$F(x) \rightarrow \Delta t = \int_t^{t+\delta t} f(s) dt = F(t + \delta t) - F(t)$$

where  $F(x)$  is a complex system during its evolution

Function synchronisation = [time, location]

$$\text{Synchronisation} = 2 \cdot (0.63 \cdot \text{location}) \cdot (0.63 \cdot \text{location})$$

The time required for an object or physical quantity to move from one location to the other can be expressed as 0 to  $1 - 1/e$  (63.2%), with the final value of  $t$  as  $1 - e^{-kt}$ . The time required for physical quantity or object to fall to the (36.8%) of its initial value when it varies with time  $t$  is  $e^{-kt}$ .

The integral function was translated into the following codes to define the fitness function to be used on the GA tool which is as follows:

```
function F = mobilitysync(pop)

%Author: Divina Melomey
%University of East London

% mobilitysync compute mobilitysync function

% this considers the use of quad function with complex values for
%limit of integration. this function computes the integral between two
%specified end points where the limits are t and change in t plus t
%in this case i

%F1 = quad('sin(z)', -1+i, 2-i) %Calculate integral in MATLAB
%'sin(z)' defines the function where z is a complex variable

% F = quad ('sin(z)', i, 1-i) % calculate integral for mobility
% synchronisation

%22/06/2009

F = quad('sin(z)', i, 1-i);
```

The simulation was defined and shown in Figure 5.8 with the 'plot function' option set to 'stopping' in order to access conditions that are likely to terminate the function as illustrate in Figures 5.9 and 5.10.



### 5.3.3 Test 3: Rastrigin's Function

#### RASTRIGIN'S FUNCTION

The function is defined by the following:

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

With the following search space of

$$-5.12 \leq x_i \leq 5.12$$

where  $x = 0$  and

$n$  = the number of variables and

$i = 1, 2, \dots, n$

The Rastrigin's function is highly multimodal which produces a large number of local minima and regularly distributed. Its global minimum occur when  $f(x) = 0$ ;  $x(i) = 0$  and  $i = 1:n$

The following codes define the Rastrigin's function which the fitness function solver runs on and is as follows:

```
function scores = rastriginsfcn(pop)
%RASTRIGINSFCN Compute the "Rastrigin" function.

% Copyright 2003-2004 The MathWorks, Inc.
% $Revision: 1.3.4.1 $ $Date: 2004/08/20 19:50:22 $

% pop = max(-5.12,min(5.12,pop));
scores = 10.0 * size(pop,2) + sum(pop.^2 - 10.0 * cos(2 * pi .*
pop),2);
```

Rastrigin's function is a typical example of a non-linear multimodal function which was first proposed by Rastrigin (Rastrigin and Erenshteyn, 1975; Torn and Zilinskas, 1989; Muhlenbein *et al.*, 1991) and is used to test the performance of GAs.

The Rastrigin's function simulation setup is indicated in Figure 5.3.

## 5.4 SIMULATION RESULTS

### 5.4.1 Results for the Mobility RMI Function

After running the simulation using '@mobilityRMI' function, the 'solver' terminated after 74 iterations with the 'minimum objective value' of 1.6 as shown in figure 5.6. The 'plot function' for the 'best fitness' and 'best individual' is illustrated in Figure 5.7. The auto-code generated can be found in Appendix A.

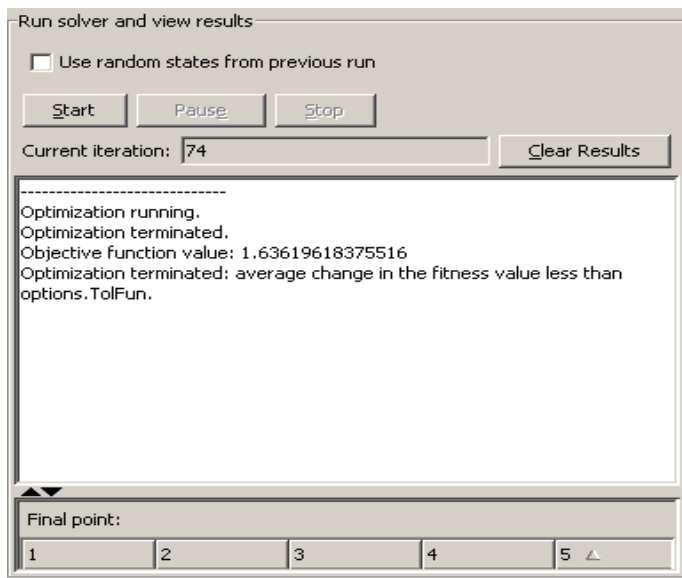


Figure 5.6: '@mobilityRMI' results

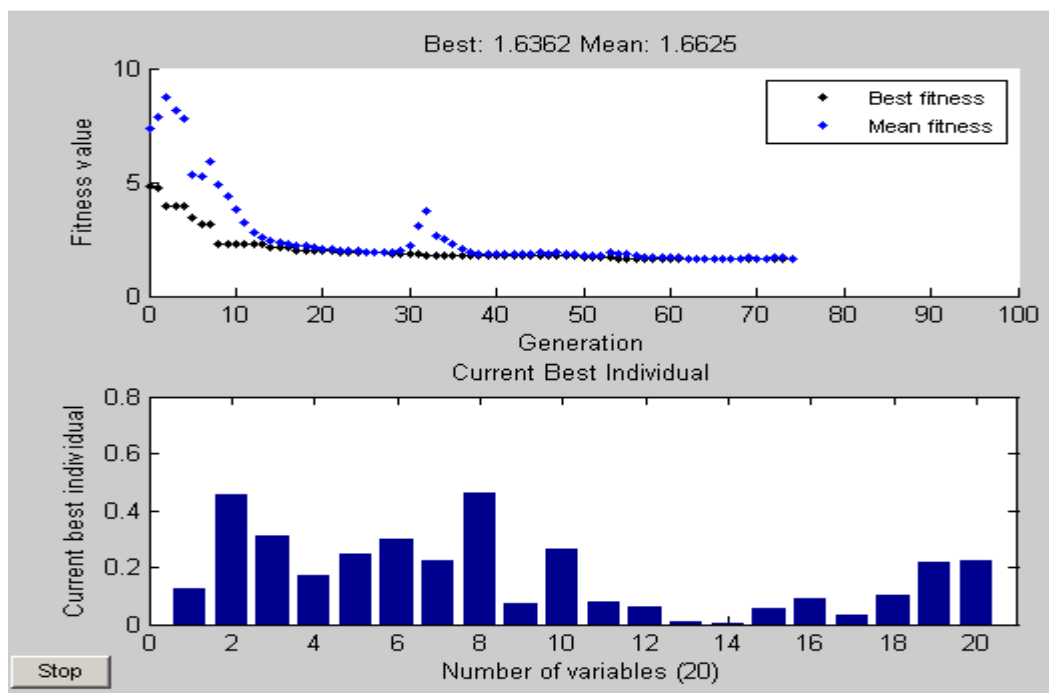


Figure 5.7 '@mobilityRMI' function plot

### 5.4.2 Results for the Mobility Synchronisation Function

After running the simulation for '@mobilitysync', the result is indicated by the visual plot in Figure 5.8 and Figure 5.9 after 51 'iterations' is given in Figure 5.10 with 'minimum objective value' of 2.12. Automatic generated code for reuse and preservation is also provided in Appendix A;

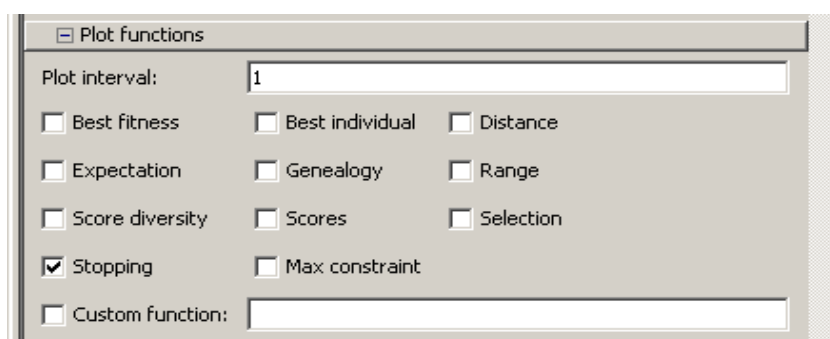


Figure 5.8 '@mobilitysync' function option

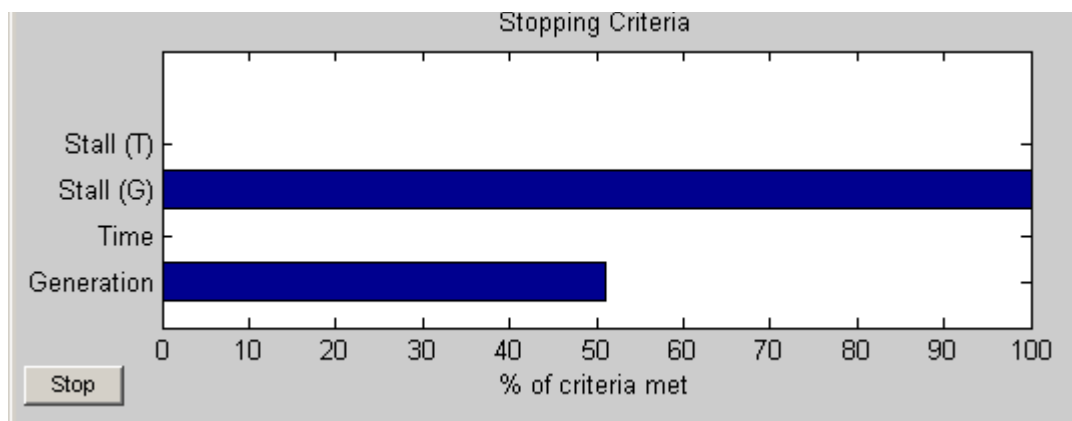


Figure 5.9: '@mobilitysync' function plot

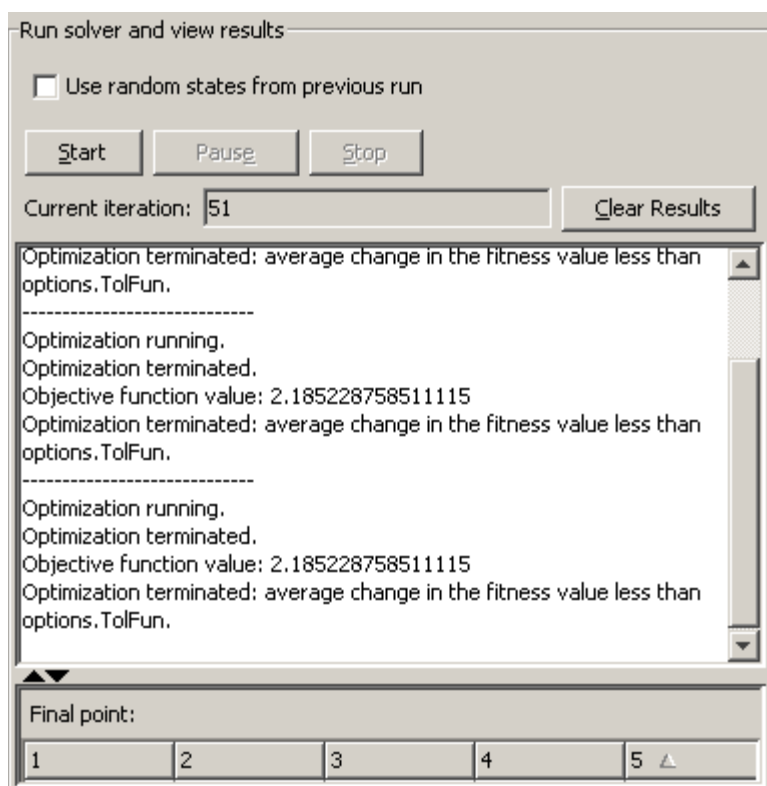


Figure 5.10: '@mobilitysync' function results

### 5.4.3 Results for Rastrigin's function

The solver '@rastriginsfcn' produced simulation results after 58 iterations. The following figures display the results as shown in Figures 5.11 and 5.12 with a 'minimum objective function' value of 38.49. 'Plot functions' selected identifies the 'best fitness' and 'best individuals'. The 'plot function' displays, monitors and outputs results visually during runtime as illustrated in Figure 5.12. The automatic code generated for Figures 5.11, 5.12 is provided in Appendix A.

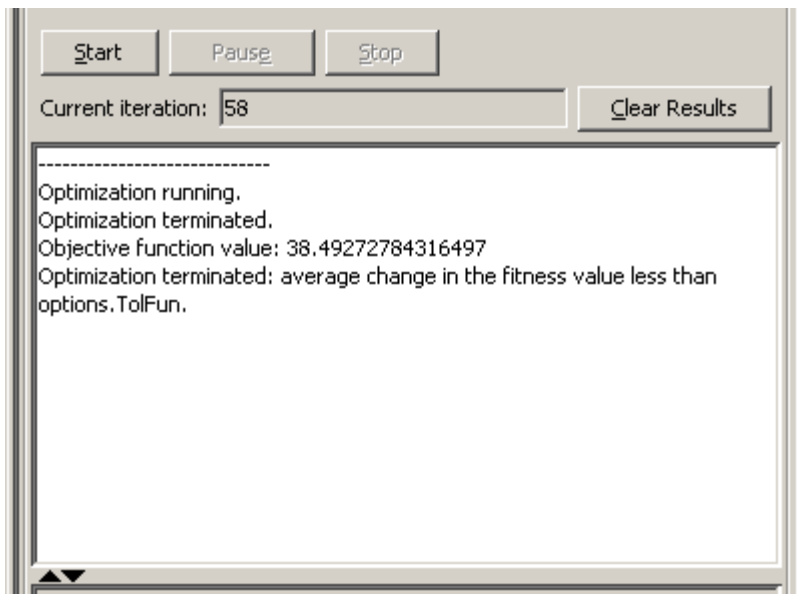


Figure 5.11: '@rastriginsfcn' simulation results

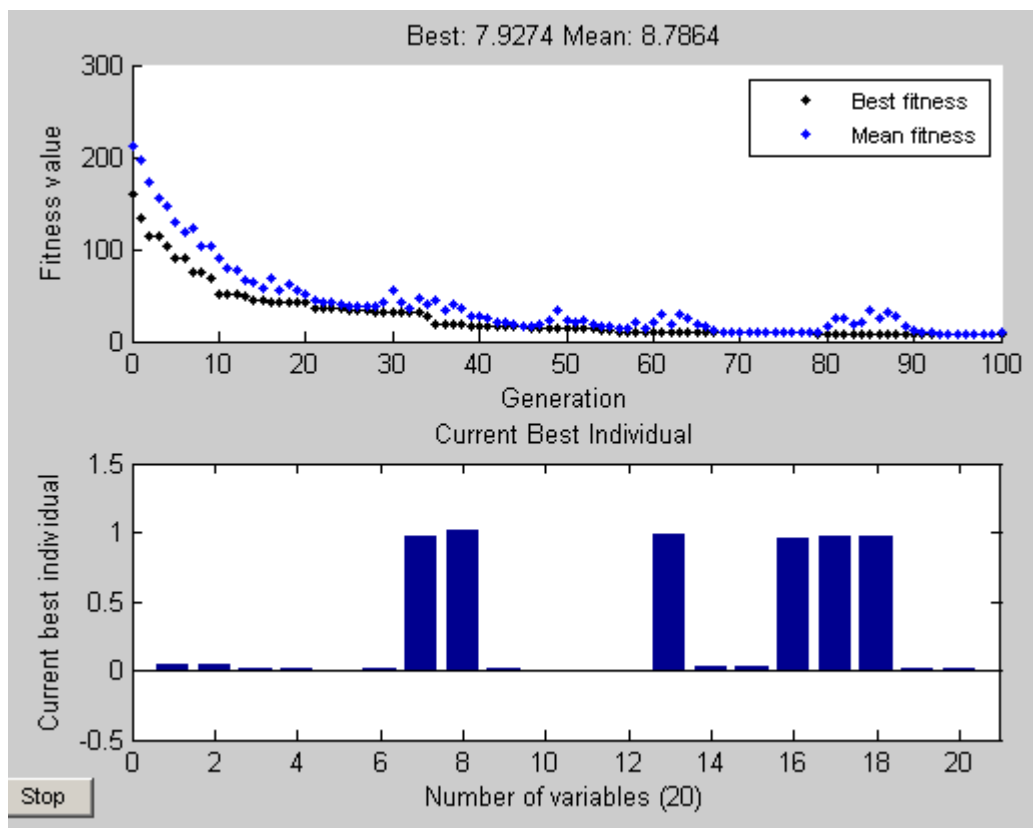


Figure 5.12: '@rastriginsfcn' simulation function plot

## 5.5 Evaluation of Simulation Results

This section provides a comparative analysis of all the results from the simulations and testing conducted using the fitness function solvers '@mobilitysync', '@mobilityRMI' and '@rastriginsfcn' to evaluate the fitness of the individual variables in a given population. The purpose of this analysis is to compare variable 'final point co-ordinates' at which the simulation is terminated. The comparison is based on three main areas and they are; the 'number of fitter individuals' at termination, the 'elite count' and the 'minimum objective function value' with respect to iterations in a given 'population'.

For each of the functions the evaluation was run at least 10 times and the average values were chosen. Table 5.1 indicates the 'final point co-ordinates' at which the simulation terminates for each of the variables for fitness functions '@mobilitysync', '@mobilityRMI' and '@rastriginsfcn' with the evaluation functions and the number of iterations at 51, 54 and 65 respectively.

<b>Final Point at which Genetic Algorithm Terminates</b>				
<b>Variable Number</b>	<b>Requirements</b>	<b>Genetic Algorithm Solver</b>		
		<b>@mobilitysync</b>	<b>@mobilityRMI</b>	<b>@Rastriginsfcn</b>
1	Abstraction	0.393	0.092	-0.034
2	Addressing	0.707	0.005	-0.036
3	Availability	0.752	0.099	-1.003
4	Calling	0.923	0.05	-0.01
5	Concurrency	0.227	0.161	0.936
6	Encoding	0.088	0.82	0.154
7	Fault Tolerance	0.266	0.078	0.958
8	Inheritance	0.199	-0.269	0.111
9	Invocation	0.497	0.056	0.994
10	Latency	0.323	0.055	0.032
11	Location	0.225	-0.06	0.017
12	Message Passing	0.833	0.111	0.001
13	Synchronisation	0.44	0.043	0.963
14	Naming	0.441	0.076	0.037
15	Openness	0.411	-0.032	0.061
16	Persistency	0.749	0.159	-1.004
17	Polymorphism	0.698	-0.246	1.042
18	Replication	0.661	0.243	1.997
19	Transparency	0.661	0.243	0.967
20	Self Protective and Certified	0.777	0.121	1.072

Table 5.1: Final Point Co-ordinates

Table 5.1 enables all the evaluation functions or fitness to be compared with the Rastrigin's function which was used to benchmark performance. At this 'final point coordinates' the 'minimum objective value' for fitness functions '@mobilitysync',

'@mobilityRMI' and '@rastriginsfcn' were 0.70, 1.04 and 24.08 respectively with the following 'function plots':

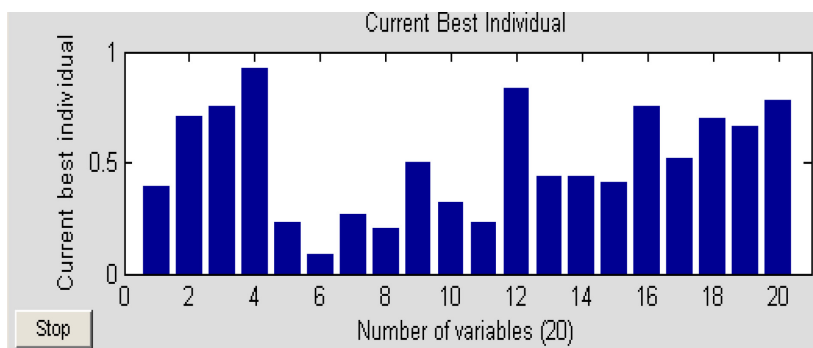


Figure 5.13: '@mobilitysync' plot function

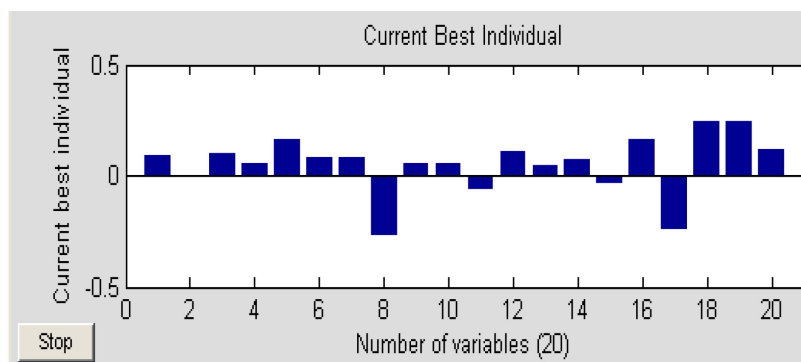


Figure 5.14: '@mobilityRMI' plot function

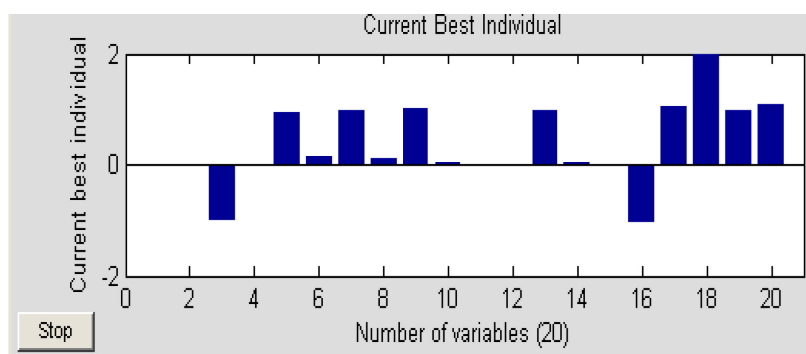


Figure 5.15: '@rastriginsfcn' plot function



The next section discusses the results of the simulation and testing of the fitness function for selecting mobility requirements.

## 5.6 Discussion of Simulation Results

This section provides a detailed discussion of the simulation test results for the fitness function solvers '@mobilitysync', '@mobilityRMI' and '@rastriginsfcn'. Figures 5.13, 5.14 and 5.15 indicate the 'number of variables' against the 'current best individual'.

Table 5.1 is the outcome of the GA evaluation of the requirement variables highlighted in columns one and two. The variables were selected from Table 4.1. The interval or search space for the fitness functions '@mobilitysync' and '@mobilityRMI' is  $-0.5 \leq x \leq 0.5$ . This means that variable whose 'final point co-ordinate' is within this search space is among the 'fittest individual' selected for the next generation. Any 'final point co-ordinate' which is outside this interval is discarded.

In benchmarking the results from mobility fitness functions with the Rastrigin's, the search space is scaled down to  $-1.0 \leq x \leq 1.0$  from its original search space of  $-5.12 \leq x \leq 5.12$  to allow for margin of error for the selection of the 'fittest individuals' to be compared with the mobility selection variables.

The 'final point co-ordinates' for fitness function @mobilitysync within the interval  $-0.5 \leq x \leq 0.5$  are Abstraction, Addressing, Concurrency, Encoding, Fault Tolerance, Inheritance, Invocation, Latency, Location, Synchronisation, Naming and Openness. These are the fittest individuals that have been selected for the next generation and the

rest are simply discarded. Results from Table 5.1 indicate that Availability, Calling Message Passing, Polymorphism, Transparency and Self Protective and Certified are all discarded.

The 'final point co-ordinates' for fitness function '@mobilityRMI' within the interval  $-0.5 \leq x \leq 0.5$  are Abstraction, Addressing, Availability, Calling, Concurrency, Fault Tolerance, Inheritance, Invocation, latency and Location and Message passing, Synchronisation, Naming, Openness, Persistency, Polymorphism, Replication, Transparency and Self Protection and Certified. These are the fittest individuals that have been selected for the next generation and the others are simply discarded. In this run and iteration the variable 'encoding' is discarded.

According to Table 5.1, Rastrigin's function's fittest individuals are Abstraction, Addressing, Calling, Concurrency, Fault Tolerance, Inheritance, Invocation, Latency and Location and Message passing, Synchronisation, Naming, Openness and Transparency. The individuals discarded are Availability, Persistency, Polymorphism, Replication, and Self Protection and Certified.

The ratio of the individual discarded for the fitness function '@mobilitysync': '@mobilityRMI': '@Rastriginfcn' is 7:1:5. This means that when the fitness functions evaluated the requirements, the fitness function '@mobilitysync' discarded more individuals during the fitness selection process than '@Rastriginfcn'. The fitness function '@mobilityRMI' only had one to discard. This means that the '@mobilitysync' rigorously sifted through all the variables that were essential for mobility development as compared to that of the '@mobilityRMI' and these results compared favourably well to the results from '@Rastriginfcn'. Furthermore, the mobility fitness functions are meant to support

the development process and not to replace the human effort. The mobility fitness function will serve as a support tool for software developers to specify and design the mobility requirements of the system to meet systems requirements more closely than human effort alone.

## **5.7 Real Life Usefulness of Results**

### **Online Banking Scenario**

The subsidiary online banking operations in remote locations across the world rely heavily on security, availability, autonomy, migration, synchronisation, persistency, concurrency, name services, transparency and fault tolerance of their processes. Migration and remote access to data for example are critical functions in their operations. This set of requirements allows bank customers to apply for accounts online, logon and manage their finances. Customers can create regular payments and standing orders for their nominated accounts, update their personal information and send secure emails to their respective banks. The outcomes of this research when applied in the above banking scenario will enable automatic selection of processes based on efficient criteria of a set of variables. This will further result in the development of better system function and customer experience. The Mobility Requirement Elicitation, Fitness Classification and Code Transformation phases also enable the capturing of core and critical requirements peculiar and unique to the distributed systems application area.

Table 4.1 highlights the requirements necessary for analysing information processes that require remote access to data and migration of similar data resource. The assigned numbers are referred to as variables which describe the following requirements; Abstraction, Addressing, Availability, Calling, Concurrency, Encoding, Fault Tolerance,

Inheritance, Invocation, Latency, Location, Message passing, Mobility/Migration, Naming, Openness, Persistency, Polymorphism, Replication, Resource sharing, Scalability, Security, Self Protection and Certified, Synchronisation and Transparency.

From Table 4.1 the simulation process begins with the problem. During the iteration, 'best fitness' and 'best individual' are monitored.

The outcome of the simulation is based on 'number of fitter individuals' selected for the banking application. In this scenario, those variables selected for the simulation are concurrency, fault tolerance, synchronisation, naming services and openness. The rest are simply discarded. This is 5 out of 10 variables previously identified by the banking application software developer. These 5 variables represent the precise fitness specification to be used in the development of the mobile agent banking application. This however does not replace entirely the developer's expert judgement but rather compliments the effort of coming out with the appropriate specification for the development. The developer benefits from using the 'number of fitter individuals' to compliment his effort in deciding on the appropriate specifications needed for developing the mobile agent banking application. To the developer, this will serve as an independent tool for addressing migration issues in mobile agent application development. Another benefit to the developer is the tool's ability to select 'number of fitter individuals' in solving complex and unpredictable problems associated with the migration of mobile agent banking application.

This process can be automated and form the basis for case tool development since this is the first time in mobile agent research that the underlying principles of Genetic Algorithms fitness functions have been used in selecting system requirements for the

development of mobility agent-based applications. This provides another benefit to the developer of the online banking scenario discussed earlier.

It is therefore recommended that both mobility function 'solvers' should be used in order to compare results for the same problem. There is a high industry demand for CASE tool environments that can effectively support the software specific process such as mobility capture in the software development process. For example, in a safety critical environment modelling mobility using MaMM will enable the mobility requirements to be captured and supported in the development environment which is a benefit that current methodologies fail to address. Current methodologies such as MaSE methodology focused on the output models of the analysis phase of systems development and failed to identify why mobility is needed and its association with the requirements of the systems (DeLoach *et al.*, 2001). Since MaSE did not recognise mobility as a requirement, it did not also consider remote method invocation, synchronisation and persistency as essential component for developing mobile agent-based systems. Also, GAIA lacks the concepts to support the modelling and reasoning of the agents' mobility and the social interaction in an environment and therefore paid less attention to the core mobility requirements of remote method invocation, persistency and synchronisation for developing mobile agent-based systems (Wooldridge *et al.*, 2000; Huang *et al.*, 2007). Furthermore, Tropos and Prometheus methodologies did not consider these core mobility requirements at all as focus were mainly on development of multi-agent systems (Perini *et al.*, 2002; Castro *et al.*, 2002; Bresciani *et al.*, 2004; Padgham and Winikoff, 2004). These methodologies were developed for non-mobile agent systems and as such the requirements selected are not essential for developing mobile agent systems. The development of MaMM using the mobility fitness function is a new contribution this research is bringing to the field. MaMM is captured in the Figure 4.2.

For the first time in mobile agent research, mobility fitness functions are used to select and evaluate mobility requirements for software development. The results from the evaluation have also proven to be more efficient at selecting requirements than human effort. Current methodologies capture mobility as a component and high level system requirement in a distributed system environment. Again, current methodologies also lack the ability to formalise and effectively support systems which put mobility as a high priority in the systems development process.

## 5.8 Summary

In this chapter the fitness function solvers for mobility were tested and were used to select mobility requirements without the intervention of the user for the first time. This is the first time in mobile agent research that a mobility methodology has enabled a tool to independently select mobility requirements. Results from the testing of fitness functions '@mobilitysync' and '@mobilityRMI' were presented and compared with the test results from the Rastrigin's function solver which was used as a benchmark to measure performance and selections of fittest individuals for mobility applications development. The results for fitness function simulation '@mobilitysync' fall within the range  $-0.5 \leq x \leq 0.5$  and that for fitness function '@mobilityRMI' fall within the range of  $-0.5 \leq x \leq 0.5$ . This means that at all times, regardless of the function 'solver' used, the fittest will always fall within the range  $-0.5 \leq x \leq 0.5$ . Based on the results displayed, the '@mobilitysync' and '@mobilityRMI' fitness functions performed comparatively better in selecting the mobility requirements when benchmarked with the well known Rastrigin's function for measuring performance and effectiveness of Genetic Algorithms.

# CHAPTER 6

## Conclusions and Future Work

### 6.1 Summary

The last few years have seen the proliferation of approaches and methodologies for modelling multi-agent systems with little attention to mobile agent systems development. This thesis addresses the lack of a mobility methodology to model mobility in mobile agent-based systems. Current developments in agent technology and pervasiveness of distributed computing have change the focus of research to mobile agent technology exploitation. There is currently no methodology for developing the mobility of the mobile agent-based systems using fitness function for the selection of mobility requirements. This thesis presents a methodology for modelling the mobility of mobile agent based systems.

The first part of this thesis solicited converging opinions from experts using Delphi Study. Case studies were also used to collect further details from experts on how and what processes were involved in different types of online applications and furthermore to evaluated the draft of the methodology. The feedback from the case studies gave an indication of how the methodology should be developed. Mobility requirements derived from both Delphi study and the case studies were simulated using fitness function modelled mathematically. The methodology developed from this thesis is known as Mobile agent-based Mobility Methodology (MaMM).

The aim of this research was to develop a methodology for modelling mobility in mobile agent-based system using a GA approach. This has been presented in MaMM which is made up of four phases. MaMM will assist developers to model mobility through all the phases of applications development. Bottom up approach was used, drawing on expertise, attributes, properties and elements embodied in Delphi study, case studies and simulation environment. The conclusions will therefore re-examine the results in terms of the success in achieving the aims and objectives identified for this research programme.

## **6.2 Delphi Study**

The Delphi study was used to solicit emerging opinions from experts and the results gave a strong indication of the types of complex case studies to select for the methodology development. The Delphi study was used to gather information relevant to the development of MaMM from experts on software development (Chapter 3).

## **6.3 Case Studies**

Case studies were used to identify mobility requirements, test and evaluate the methodology (Chapter 3). This is illustrated in Figure 1.1 which indicates how the process fed into each other from the bottom to the top. These mobility requirements were the core for developing mobile agent-based systems which were general mobility requirements and are not necessarily used for all applications as each application is unique as tabulated in Table 4.1. There are three important requirements identified from the case studies which are critical and essential to mobility on mobile agent-based platform. These requirements are remote method invocation, synchronisation and



persistence. In order to enable mobility on mobile agent platform, the core requirements for modelling mobility are modelled using the principles of Genetic Algorithm. Mobile agent applications are usually built and deployed on distributed platforms.

## **6.4 Mobile agent-based Mobility Methodology (MaMM)**

Mobility in mobile agent-based system is the process by which a mobile agent migrates from one platform or location to another autonomously. This methodology known as MaMM, models mobility by bringing together key aspects of earlier approaches for modelling mobility, which is independent of any specific approach in the applications development process. MaMM is a four phased system comprising of Mobility Requirement, Fitness Classification, Code Transformation and Mobility Implementation. This has a focus on the application of Genetic Algorithms for fitness selection of mobility requirements (Chapter 4).

Limitations in the modelling of mobile agent methodologies and approaches have been highlighted and the original contribution of MaMM has provided new insights of the process. Current methodologies such as MaSE methodology focused on the output models of the analysis phase of systems development and failed to identify why mobility is needed and its association with the requirements of the systems (DeLoach *et al.*, 2001). Since MaSE did not recognise mobility as a requirement, it did not also consider remote method invocation, synchronisation and persistence as essential component for developing mobile agent-based systems. Also, GAIA lacks the concepts to support the modelling and reasoning of the agents' mobility and the social interaction in an environment and therefore paid less attention to the core mobility requirements of

remote method invocation, persistency and synchronisation for developing mobile agent based systems (Wooldridge *et al.*, 2000; Huang *et al.*, 2007). Furthermore, Tropos and Prometheus methodologies did not consider these core mobility requirements at all as focus were mainly on development of multi-agent systems (Perini *et al.*, 2002; Castro *et al.*, 2002; Bresciani *et al.*, 2004; Padgham and Winikoff, 2004). These methodologies were developed for non-mobile agent systems and as such the requirements selected are not essential for developing mobile agent systems. The development of MaMM using the mobility fitness function is a new contribution this research is bringing to the field. MaMM is captured in the Figure 4.2. Table 6.1 recaptures the strength and limitation of each existing Multi-agent Systems methodologies together with that of the MaMM. The conceptual and abstraction levels of MaMM has been developed and rigorously tested in Chapter 4 and Chapter 5 and as such satisfies the requirement, analysis and design phases of the methodology as indicated in Table in Figure 6.1. However, though the implementation phase has been conceptually tested it has not gone through the rigour the other methodologies have undergone and as such it scores No for the implementation phase.

.

Methodology Phases	Requirement Phase	Analysis Phase	Design Phase	Implementation Phase
MaSE	No	Yes	No	No
GAIA	No	Yes	Yes	No
TROPOS	Yes	No	No	No
Prometheus	No	Yes	Yes	Yes
MaMM	Yes	Yes	Yes	No

**YES** – Strength of the Methodology **NO**- Limitation of the Methodology

Table 6.1: Multi-agent Systems Methodologies and MaMM

Mobility fitness functions were then developed from mathematical models of the mobility requirements. These requirements were based on the selected case studies that exhibit characteristics of mobile agents. The mobility requirements were then simulated using a GA tool and the various results were benchmarked against Rastrigin's function. Rastrigin's function is a widely accepted function for testing performance of Genetic Algorithms.

### 6.4.1 Simulation and Evaluation of Results

Mobility fitness functions are modelled mathematically using the core mobility requirements which were identified as remote method invocation, synchronisation and

persistence. These were translated into mobility fitness functions similar to the Rastrigin's function; Rastrigin's function was used for benchmarking performance of the mobility fitness functions. Rastrigin's function is widely used for testing performance of Genetic Algorithms and the search space is  $-5.12 \leq x_i \leq 5.12$ , however, the search space was scaled down to  $-1.0 \leq x \leq 1.0$  to allow for some margin of error for comparison in the selection of mobility variables.

The following observations were made when the outcomes of mobility fitness functions were benchmarked with the Rastrigin's functions.

The mobility fitness functions '@mobilitysync' and '@mobilityRMI' closely select the mobility requirement variable for the development of mobile agent-based systems given the 'population' and the 'number of variables'. The mobility fitness functions select requirement variables from the fitness range  $-0.5 \leq x \leq 0.5$  while Rastrigin's function selects from a range  $-1.0 \leq x \leq 1.0$ . The requirements that do not pass the fitness test are those that do not fall within the fitness range and are discarded whereas those that pass the fitness test are used to model mobility of the mobile agent in the second part of the fitness classification phase MaMM. Another important finding from the study was that the phases of MaMM are adaptable as confirmed by responses from the Delphi study.

The second phase of the MaMM which is known as 'fitness classification', analysed the need for mobility using mobility fitness classification model which groups mobility development needs in four categories. The model was developed from the outcome of the case studies based on time, behavioural, addressing and security needs. Three of

these categories must be satisfied for any of the mobility applications development. The fitness of mobility requirements were modelled mathematically based on element composition of the requirements defined and was applied at the 'fitness classification' phase of the MaMM. Two of the mobility requirements at the core of any mobility application, the remote method invocation and the synchronisation were then transformed into a fitness functions in order to select the fittest mobility requirements given the population and the number of variables for such applications. The mobility fitness function were '@mobilitysync', '@mobilityRMI' and Rastrigin's function '@Rastriginfcn' for benchmarking the mobility fitness function results. The GA tool was use to run the fitness selection which meant that whatever requirements survived into the next generation were the fittest of the requirement for further development. This means that when the mobility fitness functions evaluated the requirements, requirements that are not selected are discarded. Furthermore, the mobility fitness functions are meant to support the development process and not to replace the human effort. The mobility fitness function will serve as a support tool for software developers to specify and design the mobility requirements of the system to meet systems requirement more closely that human effort alone. This process can be automated and form the basis for case tool development since this is the first time in mobile agent research that Genetic Algorithms fitness functions have been used in selecting system requirements for mobile agent-based applications development.

Fitness functions and the GA simulation tool which utilises GA principles were also integrated into the Fitness Classification phase of MaMM to select the mobility requirement for mobile agents based on applications development. These mobility requirements were simulated on a GA simulation platform and results were

benchmarked against Rastrigin's function selection. As in any distributed systems architecture, consideration is given to existing standards such as FIPA and MASIF, hence the mobility design layer diagram in chapter 4 indicates which layer in the distributed environment or platform that the mobility applications is built on. The mobility platform layer lies on top of the distributed platform layer and above this, is the online applications layer. This serves as a guide for mobility application developers in agent-based systems.

Mobile Agent System Interoperability Facility (MASIF) and the Foundation for Physical Intelligent Agents (FIPA) are standards which provide support and management for software agents, the execution environment and resources. MASIF and FIPA are part of the Object Management Group (OMG) whose work on Mobile Agent Facility (MAF) was to promote inter-operability amongst agent platforms and to provide all interfaces between agent and agent systems. This research outcome was developed giving consideration to MASIF and MAF standards with respect to basic concerns like the agent management, migration and tracking from one platform to another. MASIF primarily identifies a distributed agent environment with reference to a place, and in this research it is known as a platform in MaMM where mobile agents visit and executes its codes. Another aspect of MASIF is the support for region or localisation of authority which is similar to a zone in MaMM in terms of providing security accesses to migration of agents within a zone. The FIPA 2000 specification is related to agent mobility, heterogeneous interaction of agents, agent based systems, communication and agent transport, which are issues not covered by MASIF, however, in this research communication issues were considered.

## 6.5 Research Contributions

This thesis makes significant contributions to the state of the art in the following ways: a Mobile agent-based Mobility Methodology (MaMM) to model mobility in a mobile agent based system. MaMM is made up of four phases which provide a guide in the formal analysis, design and implementation of mobile agent based systems.

The methodology is a guide to direct the development of solutions in respect of modelling mobility of mobile agents. The research phases proposed are not new but rather similar to incremental and developmental methodologies which have been tried and tested (Boehm 1988, Greer and Ruhe, 2004, Qui and Riesbeck, 2008). Feedback proposed between the phases at each iteration will assist in the design of systems and will be ready to face the uncertainties in complex problem domains. This is the first time that a fitness function has been used to select the requirement for developing mobility mobile agent-based application. The following are the key aspects of the contributions:

1. A Mobile agent Mobility Methodology (MaMM) which describes the phases of mobility systems development. The four phases of the MaMM are the mobility Requirement Elicitation, Fitness Classification, Code Transformation and Mobility Implementation. For the first time in mobile agent research, this work introduces Genetic Algorithm (GA) principles for selecting and assessing the fitness of the mobility elements during the Fitness Classification phase of the system development process. This involves using Genetic Algorithm principles to select, mutate and perform crossover functions on a specified number of variables in a given population. These variables and other parameters when specified using GA principles, together with the fitness function generated from the formulated

mobility function. This further provides an optimised solution for each variable when benchmarked with Rastrigin's function.

2. The Design layer diagram is a distributed mobility platform which specifies the core requirements for the development of all mobility applications. This diagram illustrates mobility requirements necessary to design online based applications. This design diagram shows how these mobility requirements were derived from the generic mobility requirement for designing distributed applications from selected online cases.
3. The Mobility Fitness Classification model. This classifies high interactive activities using a quadrant-like model in which at least three of the groupings must be satisfied in order to develop mobile agent-based online applications. The groupings are time, behavioural, addressing and security.
4. The mobility design layer diagram. This demonstrates how mobility applications can be built on distributed platform architecture. This is a three layered diagram and comprises of bottom layer, middle layer and top layer. The bottom layer is the distributed platform layer which sets the basis for building mobile applications, the middle layer is the mobility platform which specifies the mobility elements that must present in mobile applications which forms the basis for the mobility fitness functions and the top layer is the internet/online layer.
5. Fitness functions for mobility. These were modelled for the entire mobility requirements for developing mobile applications. Modelling these mathematically took into consideration the components that make up each of the mobility requirements identified.



6. Fitness functions for Genetic Algorithm. For the first time in mobile agent research, fitness functions have been modelled to select and determine the performance of mobility variables using Genetic Algorithm tool in a specified population size, in developing mobile agent applications.

It is possible for mobile agents to be modelled if interactive activities are high, for example where mobile agents are required to visit many platforms to gather and return information or to report from remote locations. Information and data therefore needs to be available and updated in a real-time fashion. MaMM is able to model migration of mobile agents to reap the benefits of mobile agents system. Some of the benefits are the reduction in the consumption of network bandwidth, reduction in latency and an increase in fault tolerance. The MaMM approach will also assist software developers to easily conceptualise solutions to complex software systems.

## 6.6 Future Work

The results from the simulation and testing demonstrate that the methodology developed is successful in the selection of mobility requirements to develop mobile agent-based systems. Possibilities for future work are:

- The automation of fitness selection process which will serve as a basis for a further research for CASE tool development for the software industry.
- The deployment of MaMM as a CASE tool in industry with automated features for fitness selection of mobility requirements in mobile agent-based software development. Automating mobility using fitness function could help software developers to specify and design the mobility requirements of the

system more closely, compared to current methodologies such as MaSE, GAIA, Prometheus and Tropos (Wooldridge *et al.*, 2000; DeLoach *et al.*, 2001; Perini *et al.*, 2002; Castro *et al.*, 2002; Bresciani *et al.*, 2004; Padgham and Winikoff, 2004; Huang *et al.*, 2007). An open end question in this regard is how much cost is involve in deploying the CASE tool in a large and complex network and other related constraints.

- The development and integration of security mechanisms as part of the developed MaMM. Security functions can also be developed to explore the strength of the MaMM from a security perspective. A reflective blind spot that was not adequately compensated for will be the introduction of additional complexities which might cause the system to fail in obscure ways or even lead to the exploitation of other vulnerabilities that might be identified. A possible research question at this point is; how can security be integrated as part of the methodology?

Finally, although the results from the research shows that GA is more efficient in selecting fitness requirements after benchmarking, further research can be conducted in the following areas;

- what are the performance issues when deployed on large systems with higher data processing requirements? Example, in handling high volume image data on high performance networks.

## References

- ADLER, M. & ZIGLIO, E. (1996) Gazing into the oracle: The Delphi Method and its application to social policy and public health, London, Jessica Kingsley Publishers, pp.3-33.
- ANDREWS, D. C. (1991) JAD: A crucial dimension for rapid applications development. *Journal of Systems Management*. Volume 42(3), pp. 23-31.
- ARFKEN, G. (1985) The Gamma Function (Factorial Function), Orlando, FL: Academic Press, pp 339-341 and pp 534-572.
- ARLOW, J., QUINN, J. & EMMERICH, W. (1999) Literate Modelling- Capturing Business Knowledge with UML IN BEZIVIN, J. & MULLER, P. A. (Eds.) *Proceedings of UML'98, Lecture Notes in Computer Science (LNCS)* Mullhouse France, Volume 1618, Springer Verlag, pp.165-172.
- BÄCK, T. (1996) *Evolutionary Algorithms in Theory and Practice*, New York Oxford University Press.
- BAKER, J. E. (1985) Adaptive Selection Methods for Genetic Algorithms *Foundation of Genetic Algorithms (FGA1)* pp. 101-111.
- BALDI, M., GAI, S. & PICCO, G. P. (1997) Exploiting Code Mobility in Decentralized and Flexible Network Management *Proceedings of the First International Workshop on Mobile Agent*, Berlin, Germany , pp. 13-26.
- BALMELLI, L. (2007) An Overview of Systems Modeling Language for product and Systems language for product and systems development. *Journal of Object Technology*, Volume 6(6), pp. 149-177.
- BANCROFT, M. & AL-DABASS, D. (2004) A Combat Simulation Aid for Dungeon and Dragons. *Proceedings of 5th Game-On International Conference*. Vol. 1&2, Reading, UK, pp.61- 72.
- BAUER , B. (1999) Extending UML for the Specification of Interaction Protocols. *Proceedings of the First International Workshop, AOSE 2000, Lecture Notes in Computer Science (LNCS)*, Volume 1957, Limerick, Ireland, pp121-140.
- BAUER, B. (2001) Agent UML A Formalism for Specifying Multiagent Software Systems. *International Journal of Software Engineering and Knowledge Engineering* 11(3), pp. 207-230.
- BAUER, B. & ODELL, J. (2005) UML 2.0 and agents: how to build agent-based systems with the new UML standard. *Journal of Engineering Applications of Artificial Intelligence*, Volume 18 (2), pp. 141-157.
- BAUMEISTER, H., KOCH, N., KOSIUCZENKO, P. & WIRSING, M. (2003) Extending Activity Diagrams to Model Mobile Systems. In Aksit, M., Mezini, M. & Unland, R. (Eds.) *International Conference NetObjectDays, NODE 2002. LNCS*. Erfurt, Germany.

- BELL, W. (1997) Foundations of Futures Studies: Human Science for a New Era. History, Purposes, and Knowledge, New Brunswick (NJ): Transaction Publishers, Volume 1.
- BELLONI, E. & MARCOS, C. (2004) MAM-UML: An UML Profile for the Modeling of Mobile-Agent Applications. In proceedings of 24th International Conference of the Chilean (SCCC 2004), Lecture Notes in Computer Science, No.350, Arica, Chile.
- BLACKWELL, L., VON KONSKY, B. & ROBEY, M. (2001) Petri Net Script: A Visual Language for Describing Action, Behaviour and Plot. In proceedings of Australasian Computer Science Conference (ACSC '01), pp 29-37, Gold Coast, Qld., Australia, pp.3-13.
- BOEHM, B. W. (1988) A Spiral Model of Software Development and Enhancement. IEEE Computer, pp. 61–72.
- BONNER, M., MAYER, S., RAGGL, A. & SLANY, W. (1995) FLIP++: A Fuzzy Logic Inference Processor Library. Fuzzy Logic in Artificial Intelligence. Proceedings of The 1995 International Joint Conference on AI, LNCS, Montreal, Canada, Springer, pp.66-76.
- BOOTH, S. & HULTEN, M. (2003) Opening Dimension of Variation: An empirical study of learning in a web-based discussion. Instructional Science Volume 36, No.1&2, pp. 65-86.
- BRAZIER, F., DUNIN-KEPLICZ, B., JENNINGS, N. R. & TRUER, T. (1995) Formal Specification of multi-agent systems: a real world case. Proceedings of the first International Conference on Multi-agent Systems (ICMAS-95), San Francisco. USA, pp. 25-32.
- BRESCIANI, P., GIOGINI, P., GIUNCHIGLIA, F., MYLOPOULOS, J. & PERINI, A. (2004) TROPOS: An Agent –Oriented Software Development Methodology. Journal of Autonomous Agents and Multi-Agent Systems, Vol. 8(3), pp. 203-236.
- BRESCIANI, P. & SANNICOLÒ, F. (2002) Requirement Analysis in TROPOS: a Self Referencing Example. In proceedings of the NODE 2002 agent related Conference on Agent Technologies, Infrastructures, Tools, and Applications for e-Services. J. Miller, H. Tianfield, R. Unland, R. Kowalszyk ed., Lecture Notes in Computer Science, Springer-Verlag, Erfurt, Germany, pp. 21-35.
- BRUSTOLONI, J. C. (1991) Autonomous Agents: Characterization and Requirements, Carnegie Mellon Technical Report CMU-CS-91-204. Pittsburgh, Carnegie Mellon University
- CAIRE, G., COULIER, W., GARIJO, F., GOMEZ-SANZ, J., PAVON, J., KEARNEY, P. & MASSONET, P. (2004) MESSAGE: A Methodology for the Development of Agent-Based Applications, Springer, Lecture Notes in Computer Science, No.3155, Springer-Verlag, pp.547-59 .

- CASTRO, J., KOLP, M. & MYLOPOULOS, J. (2002) Towards Requirements-Driven Information Systems Engineering: The Tropos Project. In Information Systems, Elsevier, Amsterdam, The Netherlands, , Volume 27, pp. 365-389.
- CERVENKA, R., AND TRENANSKY, I. & CALISTI, M. (2005) Modeling Social Aspects of Multiagent Systems. The AML Approach IN MULLER, J. P. & ZAMBONELLI, F. (Eds.) The Fourth International Joint Conference on Autonomous Agents & Multi Agent Systems ( AAMAS 05). Workshop 7 : Agent –Oriented Software Engineering (AOSE), Universiteit Utrecht, The Netherlands, pp. 85-96.
- CERVENKA, R. & TRENCANSKY, I. (2004) Agent Modeling Language, Language specification. Whitestein Technologies AG, Technical report, Version 0.9.
- CERVENKA, R. & TRENCANSKY, I. (2007) Agent Modeling Language (AML): A Comprehensive Approach to Modeling Multi Agent System, Basel, Birkhauser and Informatica, Volume 29 pp. 391-400.
- CERVENKA, R., TRENCANSKY, I., CALISTI, M. & GREENWOOD, D. (2005) AML Agent Modeling Language . Towards Industry Grade Agent –Based Modeling. In Odell, J., Giorgini, P., Muller, J. P. & (Eds.) Proceedings of Agent Oriented Software Engineering V: 5th International Workshop, AOSE 2004, New York, USA, Springer-Verlag, pp. 31-46.
- CHEN, S.-C., LI, S.-T. & SHYU, M.-L. (2003) Model-Based System Development for Asynchronous Distance Learning. International Journal of Distance Education Technologies, Volume 1, pp. 39-54.
- CHHETRI, M. B., PRICE, R., KRISHNASWAMY, S. & LOKE, S. W. (2006) Ontology-Based Agent Mobility Modelling. Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), Hawaii, pp.45-54.
- CHRISTIAN, F. (1989) A Probabilistic Approach to Distributed Clock Synchronization. Distributed Computing Volume 3, pp 146-158.
- CONWAY, J. H. & GUY, R. K. (1996) Choice Numbers. In the Book of Numbers, New York, Springer-Verlag, , pp.266-267, 274, .
- COULOURIS, G., DOLLIMORE, J. & KINDBERG, T. (2005) Distributed Systems, Concepts and Design, Addison-Wesley Publishers.
- CUBALESKA, B. & SCHNEIDER, M. (2002) Detecting DoS Attacks in Mobile Agent Systems and using Trust Policies for their Prevention. In proceedings of the 6<sup>th</sup> World Multiconference Systemics, Cybernetics and Informatics, Volume 4, Orlando, USA, pp.177-184.
- CUSTER, R. L., SCARCELLA, J. A. & STEWART, B. R. (1999) The modified Delphi technique: A rotational modification. Journal of Vocational and Technical Education, 15 (2), pp. 1-10.
- CYPHERT, F. R. & GANT, W. L. (1971) The Delphi technique: A case study. Phi Delta

Kappan, 52, pp. 272-273.

CYSNEIROS, L. M., WERNECK, V. & YU, E. (2005) Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar The Journal of Computer Science and Technology, Special Issue on Software Requirements Engineering, Volume 5(2), pp. 71-79 .

DALKEY, N. C. & HELMER, O. (1963) An experimental application of the Delphi method to the use of experts. Management Science, 9(3), pp. 458-467.

DELBECQ, A. L., VAN DE VEN, A. H. & GUSTAFSON, D. H. (1975) Group techniques for program planning, Glenview, IL: Scott, Foresman, and Co.

DELOACH, S. A. (2004) The MaSE Methodology. In Federico; Gleizes, M.-P. Z., Franco (Eds.) In Methodologies and Software Engineering for Agent Systems, The Agent-Oriented Software Engineering Handbook Series : Multiagent Systems, Artificial Societies, and Simulated Organizations Kluwer Academic Publishing, pp.35-46.

DELOACH, S. A. (2006) Multiagent Systems Engineering of Organization-based Multiagent Systems. 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05). LNCS Vol 3914,. St. Louis, MO., Springer, pp.109-125.

DELOACH, S. A., WOOD, M. F. & SPARKMAN, C. H. (2001) Multiagent Systems Engineering. The International Journal of Software Engineering and Knowledge Engineering, Volume 11(3).

DEPKE , R., HECKEL , R. & KUESTER, J. M. (2001) Roles in Agent Oriented Modeling. International Journal of Software Engineering and Knowledge Engineering, 11, 281-302.

DIGNUM, V., MEYER, J. J., DIGNUM, F. & WEIGAND, H. (2003) Formal Specification of Interaction in Agent Societies. Formal Approaches to Agent-Based Systems (FAABS), Lecture Notes in Artificial Intelligence, Springer-Verlag, Volume 2699.

DURKEE, D., BRANT, S., NEVIN, P., ODELL, A., WILLIAMS, G., MELOMEY, D., ROBERTS, H., IMAFIDON, C., PERRYMAN, R. & LOPES, A. (2009) Implementing e-learning and Web 2.0 innovation: Didactical scenarios and practical implications Industry and Higher Education Volume 23, Number 4, pp. 293-300.

EID, M., ARTAIL, H., KAYSSI, A. & CHEHAB, A. (2005) Trends in Mobile Agent Applications. Journal of Research and Practice in Information Technology, Volume 37 No.4, pp. 331-351.

ERRFMEYER, R. C., ERFFMEYER, E., LANE, I. M. & (1986) The Delphi Technique: An Empirical Evaluation of the Optimal Number of Rounds Group & Organization Studies, 11, pp.120-128.

FIPA <http://www.fipa.org>.

- FLETCHER, M. (2007) Evaluating the Prometheus methodology through a case study on designing an agent-based holonic control system. *International Journal of Manufacturing Research*, Volume 2, pp. 342 – 361.
- FOGEL, D. B. (1995) *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, Piscataway, NJ, IEEE Press.
- FOSTER, I., KESSELMAN, C., NICK, J. & TUECKE, S. (2002) Grid Services for Distributed System Integration Computer. *IEEE*, Volume 35, No.6, pp. 37-46.
- FOX, G. (2003) *Messaging Systems: Parallel Computing the Internet and the Grid*. Proceedings of 10th European Parallel Virtual Machine/Message Passing Interface (PVM/MPI) User's Group Meeting, LNCS 2840, Venice, Italy, pp.1-9.
- FRANKLIN, S. & GRAESSER, A. (1996) Is It an Agent, or just a Program?: A Taxonomy for Autonomous Agents Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages. *Lecture Notes in Computer Science* 1193, Springer-Verlag, pp. 21-35.
- FUKUTAKE, H., OKADA, Y., NIJIMA, K., & (2004) 3D Visual Component based voice on input/output interfaces for interactive graphic applications. Proceedings International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004), Microsoft Campus, Reading, UK, pp.216-220.
- GALLI, D. L. (1999) *Distributed Operating Systems: Concepts and Practice*, Prentice Hall.
- GARCIA-OJEDI, J. C. & ARENAS, A. E. (2004) Extending the Gaia Methodology with Agent UML. Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, USA, pp.1456-1457.
- GARZETTI, M., GIOGINI, P., MYLOPOULOS, J. & SANNICOLÒ, F. (2002) Applying Tropos Methodology to a real case study: Complexity and Criticality analysis. Italian workshop on "Dagli OGGETTI agli AGENTI - Dall'informazione alla Conoscenza (WOA02)", Milano, Italy, pp. 7-13.
- GAVRILOVSKA, A., KUMAR, S., SUNDARAGOPALAN, S. & SCHWAN, K. (2005) Platform Overlays: Enabling In-Network Stream Processing in Large-scale Distributed Applications. In Proceedings of 15th International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV Communications of the ACM : Skamania, Washington, pp.171-176.
- GINSBURG, L. (1998) Integrating Technology into Adult Learning. *Technology, Basic Skills, and Adult Education: Getting Ready and Moving Forward*, Information Series, pp.37-46.
- GOLDBERG, D. E. (1989) *Genetic Algorithm in Search, Optimization and Machine Learning*. Machine Learning New York Addison-Wesley.

- GOLDBERG, D. E. & DEB, K. (1991) A Comparative Analysis of Selection Schemes Used in Genetic Algorithms Foundation of Genetic Algorithms (FGA1), Volume 1 pp 69-93.
- GRAY, R. S. (1995) Agent Tcl: A transportable agent system. Proceedings of the CIKM Workshop on Intelligent Information Agents, Baltimore, USA, pp 42-48.
- GREER, D. & RUHE, G. (2004) Software release planning: an evolutionary and iterative approach. . Information and Software Technology Volume 46 Pages 243-253.
- GRIMM, R., KRIMMER, R., MEIßNER, N., REINHARD, K., VOLKAMER, M., WEINAND, M. & HELBACH, J. (2007) Security Requirements for Non-political Internet Voting. IN proceedings of the 2<sup>nd</sup> International workshop on Electronic Voting. Lecture Notes in Informatics 86 pp. 203-212.
- GUSELLA, R. & ZATTI, S. (1989). The Accuracy of Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD IEEE Transactions on Software Engineering, Volume 15(7) pp.847-853.
- GUSTAFSON, D. H., SHUKLA, R. K., DELBECQ, A. & WALSTER, G. W. (1973) A comparison study of differences in subjective likelihood estimates made by individuals, interacting groups, Delphi groups and nominal groups. Organizational Behavior and Human Performance, 9(2), pp. 280 - 291.
- GÜTL, C., PIVEC, M., TRUMMER, C., GARCÍA-BARRIOS, V. M., MÖDRITSCHER, F., PRIPFL, J., & UMGEHER, M. (2005) AdeLE (Adaptive e-Learning with Eye-Tracking): Theoretical Background, System Architecture and Application Scenarios. European Journal of Open, Distance and E-Learning (EURODL), Issue 2005/II.
- HAUPT, R. L. & HAUPT, S. E. (1998) Practical Genetic Algorithms, New York, NY, John Wiley & Sons.
- HEATON-SHRESTHA, C., EDIRINGHA, P., BURKE, L. & LINSEY, T. (2005) Introducing a VLE into campus-based undergraduate teaching: Staff Perspectives on its impact on teaching. International Journal of Educational Research Vol. 43, pp 670-386.
- HENNING, M. (1998) Binding, Migration, and Scalability in CORBA. Communications of the ACM, Volume 41, No.10, pp.62-71.
- HERRIOTT, R. E. & FIRESTONE, W. A. (1983) Multisite Qualitative Policy Research: Optimizing Description and Generalizability. Educational Research, 12 pp. 14–19.
- HOLLAND, J. (1975) Adaptation in Natural and Artificial Systems, The MIT Press.
- HSU, C.-C. & SANDFORD, B. A. (2007) The Delphi Technique: Making Sense Of Consensus Practical Assessment, Research & Evaluation, Volume 12, No.10, pp.1-8.



- HUANG, W., EL-DARZI, E. & JIN, L. (2007) Extending the Gaia Methodology for the design and development of agent-based software systems. Proceedings of the 31st Annual IEEE International Computer Software and Applications Conference (COMPSAC 2007), Peking (Beijing), China, pp. 159-165.
- HUGET, M.-P. (2005) Modeling Languages for Multi-agent Systems. Proceedings of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE-2005) at AAMAS 2005., Volume 3950, Utrecht, The Netherlands, pp.16-27.
- IEEE STD 830-198 (1984) IEEE Recommended Practice for software Requirements Documentations Specification.
- INTERNET POLICY INSTITUTE (2001) Report of the National Workshop on Internet Voting. Maryland, University of Maryland, USA.
- ISENSEE, S. & RUDD, J. (1966) The Art of Rapid Prototyping, International Thomson Computer Press, London.
- JACOBSON, I., BOOCH, G. & RUMBAUGH, J. (1998) The Unified Software Development Process, Addison Wesley.
- JANSEN, W. (2002) Intrusion Detection with Mobile Agents. Computer Communications, Special Issue on Intrusion Detection Systems, Volume 25(15), pp.1392-1401.
- JANSEN, W. & KARYGIANNIS, T. (1999) Mobile Agent Security, National Institute of Standards and Technology (NIST) special publication 800-19.
- JENNINGS, N. (2000) Building Complex Software Systems: The Case for an Agent-Based Approach Communications of the ACM, Volume 44(4), pp.35-41.
- JENNINGS, N. R & Wooldridge, M (2000) Agent-Oriented Software Engineering. Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99), Lecture Notes in Computer Science 1647 , Valencia, Spain, Springer, pp.1-7.
- JENNINGS, N. R. (1999) Agent Oriented software Engineering. Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: MultiAgent System Engineering, LNAI 1647, Valencia , Spain, pp.1-7.
- JENNINGS, N. R. (2000) On Agent-Based Software Engineering Artificial Intelligence Journal 117 (2), pp.277-296.
- JENNINGS, N. R., SYCARA , K. & WOOLDRIDGE , M. (1998) A Roadmap of Agent Research and Development. International Journal of Autonomous Agents and Multi-Agent Systems Volume 1, pp. 7-38.
- JUAN, T., PEARCE, A. & STERLING, L. (2002) ROADMAP: Extending the Gaia Methodology for Complex Open Systems. In Proceedings of the first international joint conference on Autonomous agents and Multi-agent systems (AAMAS2002), Bologna, Italy, pp.3-10.

- JUDD, R. C. (1972) Use of Delphi methods in higher education. *Technological Forecasting and Social Change*, 4 (2), pp. 173-186.
- KANG, M., WANG, L. & TAGUCHI, K. (2004) Modelling Mobile Agent applications in UML 2.0 Activity Diagrams. *Proceedings of 6th International Conference on Enterprise Information Systems (ICEIS 2004)*, Porto, Portugal, pp.31-46.
- KINNY, D., GEORGEFF, M. & RAO, R. (1993) A Methodology and Modelling Technique for Systems BDI agents. IN VAN DE VELDE, W. & PERRAM, J. W. (Eds.) *Agent Breaking away : proceedings of the Seventh European Workshop and on modelling autonomous agents in Multiagents world*, LNAI1038. Berlin, Germany, Springer-Verlag, pp.56-71.
- KLEIN, C., RAUSCH, A., SIHLING, M. & WEN, M. (2001) Extension of the Unified Modeling Language for Mobile Agents. IN HALPIN, K. S. A. T. (Ed.) *In Unified Modeling Language: Systems Analysis, Design and Development Issues*, pp116-128. Idea Group Publishing Book.
- KOHN, R. (2005) Transparent Mobility in Mobile IPv6: An Experience Report. *Journal of Computer Science and Technology*, Volume 5(4), pp.1-5.
- KOSIUCZENKO, P. (2003) Sequence Diagrams for Mobility IN KROGSTIE, J. (Ed.) *Advanced Conceptual Modeling Techniques: ER 2002 Workshops, ECDM, MoblMod, IWCMQ, and eCOMO*, LNCS 2784,.Tampere, Finland, Springer, Berlin, pp. 147-158.
- KOTAY, K. & KOTZ, D. (1994) Transportable Agents. IN YANNIS LABROU AND TIM FININ, E. (Ed.) *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM), Workshop on Intelligent Information Agents*, Garthersburg, Maryland, Springer-Verlag New York, LLC, pp.447-455.
- KREMER, R. (1998) Visual Languages for Knowledge Representation. In *proceedings of Eleventh Workshop on Knowledge Acquisition, Modeling and Management*, Alberta, Canada, pp.124-129.
- KUHN, P. (2007) Closer view of Health. *Advances for Health Information Executives*. Volume 11, pp 33.
- LAMPORT, L. (1978) Time, Clock and Ordering of Events in Distributed Systems. *Communications of ACM* Volume 21(7), pp.558-565.
- LAMPORT, L. M.-S., P. M. (1985) Synchronizing Clocks in the Presence of Faults. *Journal of ACM* Volume 32, pp.52-78.
- LANGE, D. B., OSHIMA, M., G, K. & KOSAKA, K. (1997) Aglets: Programming Mobile Agents in Java. IN MASUDA , T. (Ed.) *Proceedings of Worldwide Computing and Its Applications, (WWCA'97) Lecture Notes in Computer Sciences 1274*,Tsukuba, Japan, Springer Verlag, pp.253-266.
- LARRANAGA, P., KUIJPERS, C., MURGA, R., INZA, I. & DIZDAREVIC, S. (1999)

Genetic algorithms for the traveling salesman problem: A review of representations and operators. *Artificial Intelligence Review*. 13, pp 129–170.

- LEVESQUE, H. J. (1984) Foundations of a functional approach to knowledge representation *Artificial Intelligence Journal*, Volume 23, pp. 155-212.
- LIMA, C., SASTRY, K., GOLDBERG, D. E., LOBO, F. & (2005) Combining competent crossover and mutation operators: A probabilistic model building approach. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*. New York, NY, USA.
- LIMA, E. F. A., MACHADO, P. D. L., SAMPAIO, F. R. & FIGUEIREDO, J. C. A. (2004) An approach to Modelling and Applying Mobile Agent Design Patterns. *ACM Software Engineering Notes*, Volume 29, No.3, pp. 1-8.
- LINSTONE, H. A. & TUROFF, M. (1975) *The Delphi Method. Techniques and Applications*. , Addison-Wesley.
- LOUKIL, A., HACHICHA, H. & GHEDIR, K. (2009) MA-UML: a conceptual approach for mobile agents' modelling. *International Journal of Computer Science and Network Security*, Volume 3(2-3), pp.277-305.
- LUDWIG, B. (1997) Predicting the future: Have you considered using the Delphi methodology?. *Journal of Extension*, 35 (5), pp. 1-4.
- MALINS, J. & PIRIE, I. (2005) Developing a Virtual Learning Environment for Art and Design: A Constructivist Approach. *European Journal of Higher Arts Education*, European League of Institutes of the Arts.
- MARQUES, P., SIMÕES, P., SILVA, L. M., BOAVIDA, F. & SILVA, J. G. (2000) Mobile Agent Systems: From Technology to Applications. *Proceedings of Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA 2000) Workshop on Experiences with Autonomous Mobile Objects and Agent Based System*. Minneapolis, USA.
- MAUCO, M. V., RIESCO, D. & GEORGE, C. (2001) Using a scenario model to derive the functions of a formal specification 8th Asia-Pacific Software Engineering Conference (APSEC 2001), Macau, China, pp.329-332.
- MELOMEY, D. (2006) Evaluating the Security of Mobile Agent Platforms. In *proceedings 1<sup>st</sup> Annual Conference on Advances in Computing and Technology (ACT 2006.)* School of Computing and Technology. East London, United Kingdom, pp. 48-54.
- MELOMEY, D. (2007) A Comparative Study on Modelling language in Agent Systems. . 23rd British Colloquium for Theoretical Computer Science. Report and abstracts: Bulletin of the EATCS. Oxford University Number 92, pp. 165-186.
- MELOMEY, D. (2008) Mobility Challenges in Online Application Development. 24th British Colloquium for Theoretical Computer Science. Report and abstracts: Bulletin of the EATCS, Number 95, pp. 262-263.

- MELOMEY, D., IMAFIDON, C. & WILLIAMS, G. (2007) A Comparative Study of Modelling Languages for Agent Systems. Systems and Information Science Notes (SISN) Volume 1, pp. 207-212.
- MELOMEY, D. & MOURATIDIS, H. (2006) Evaluating the Security of Mobile Agent Platforms. Proceedings of Advances in Computing and Technology Conference AC&T 2006). Crown Plaza, London, UK.
- MELOMEY, D., MOURATIDIS, H. & IMAFIDON, C. (2007). An Evaluation Current Approaches for Modelling Mobility of Agent. Proceedings of 2nd Annual Conference on Advances in Computing and Technology (ACT'2007). London, United Kingdom, pp. 71-78.
- MELOMEY, D., WILLIAMS, G. & IMAFIDON, C. (2007) Mobility Requirements on Game Platforms: An Agent Perspective. In proceedings of 11th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia & Serious Games. , University of La Rochelle, La Rochelle, France, pp. 162-167.
- MELOMEY, D., WILLIAMS, G., IMAFIDON, C. & PERRYMAN, R. (2008) A Fitness Function for Capturing Mobile Agent Mobility on Games Platform 12th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia & Serious Games. . Louisville, Kentucky, USA, pp.71-76.
- MELOMEY, D., WILLIAMS, G., IMAFIDON, C. & PERRYMAN, R. (2008) Modelling Mobile Agent Mobility in Virtual Learning Environment (VLE) using Fitness function. In proceedings of International Conference on Student Mobility and ICT. University of Maastricht, Maastricht, Netherlands, pp. 159-165.
- MILOJICIC, D., KOTZ, D., LANGE, D., PETRIE, C. & RYGAARD, C. (1999) Mobile agent applications. IEEE Concurrency.
- MOCKAPETRIS, P. & DUNLAP, K. (1988) Development of the Domain Name System. Communications of ACM SIGCOM, pp. 123-133.
- MONARCH, I. & WESSEL, J. (2005) Autonomy and Interoperability in System of Systems Requirements Development. In the proceedings of 16th International Symposium on Software Reliability Engineering (ISSRE 2005), IEEE Computer Society 2005., USA.
- MOURATIDIS, H., ODELL, J. & MANSON, G. (2002) Extending the Unified Modeling Language to model Mobile Agents. Proceedings of the Agent Oriented Methodologies Workshop (at the OOPSLA 2002), Seattle - USA.
- MÜHLENBEIN, H., SCHOMISCH, D. & BORN, J. (1991) The Parallel Genetic Algorithm as Function Optimizer. Parallel Computing, 17, pp. 619-632.
- MURATA, T., ISHIBUCHI, H. & TANAKA, H. (1996) Multi-objective genetic algorithm and its application to flowshop scheduling Computers and Industrial Engineering 30, pp. 957-968.
- NEEMA, S., SZTIPANOVITS, J., KARSAL, G. & BUTTS, K. (2003) Constraint-

Based Design-Space Exploration and Model. Synthesis. Proceedings of Third International Conference on Embedded Software (EMSOFT), Lecture Notes in Computer Science (LNCS) 2855 Philadelphia, PA, USA.

OBJECT MANAGEMENT GROUP (2000) Persistence Object Service Stand-alone document.

OBJECT MANAGEMENT GROUP'S (OMG) (2000) Mobile Agent Facility (MAF).

ODELL, J. (2002) Objects and Agents Compared. Journal of Object Technology, Volume1, No.1, pp. 41-53.

ODELL, J., PARUNAK, H. & BAUER, B. (2000) Extending UML for Agents. Proceedings of Agent-Oriented Information systems Workshop at the 17th National Conference on Artificial Intelligence (AAAI-00). Austin, Texas, USA, pp.3-17.

OMICINI, A. (2001) SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems. Proceedings of the 1st International Workshop, AOSE 2000 on Agent-Oriented Software Engineering. Limerick, Ireland, pp. 185-193.

OVERMARS, M. (2004) Game Design in Education. Proceedings International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004) Reading, UK, pp.193-200.

PADGHAM, L. & WINKOFF, M. (2002) Prometheus: A Methodology for Developing Intelligent Agents. Proceedings of the Third International Workshop on Agent Oriented Software Engineering, at AAMAS 2002, LNCS, Springer, Heidelberg, vol. 2585, Bologna, Italy, pp. 174-185.

PADGHAM, L., WINKOFF, M. & POUTAKIDIS, D. (2005) Adding debugging support to the Prometheus methodology Engineering Applications of Artificial Intelligence, Agent-oriented Software Development, Volume 18, pp. 173-190.

PAJO, K. & WALLACE, C. (2001) Barriers to the uptake of web-based technology by university teachers. Journal of Distance Education Volume 16, pp. 70-84.

PERINI, A., BRESCIANI, P., GIUNCHIGLIA, F. & MYLOPOULOS, J. (2002) Towards An agent Oriented approach to Software Engineering. Technical Report DIT-02-0015 University of Trento, Italy.

POGGI, A., RIMASSA, G., TURCI, P., ODELL, J., MOURATIDIS, H. & MANSON, G. (2004) Modeling Deployment and Mobility in Multiagent Systems using AUML in Agent Oriented Software Engineering IV. In Giorgini, P., Muller, J. P. & Odell, J. (Eds.) Lecture Notes in Computer Science 2935. Springer-Verlag

POSTEL, J. & ANDERSON, C. (1994) White Pages Meeting Report.

QIU, L. & RIESBECK, C. (2008) An Incremental Model for Developing Educational Critiquing Systems: Experiences with the Java Critiquer. Journal of Interactive Learning Research, 19 pp. 119-145.

- RAHWAN, I., JIAN, T. & STERLING, L. (2003) Integrating Social Modelling with Agent Interaction through Goal Oriented Analysis. *Computer Systems Science and Engineering*, special issue; *Software Engineering for Multi agent systems*.
- RAO, B. R. (1995) Making the Most of Middleware. *Data Communications International* Volume 24 pp. 89-96.
- RASHID, A. & CHITCHYAN, R. (2003) Persistence as an Aspect. *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development* Boston, Massachusetts, pp. 120-129.
- RASTRIGIN, L. & ERENSHTEYN, R. (1975) The Group of Algorithms. *Proceedings of IV International Joint Conference of Artificial Intelligence*. Tbilisi, USSR, vol. 3 pp. 138-144.
- RAY, S. S., BANDYOPADHYAY, S. & PAL, S. K. (2005) New operators of genetic algorithms for traveling salesman problem: : Application to Microarray Gene Ordering Berlin, Springer Berlin / Heidelberg.
- ROACH, M. P. & STILES, M. J. (1998) COSE - A Virtual Learning Environment founded on a Holistic Pedagogic Approach. *CTI: Software for Engineering Education*, No 14, pp.5-11.
- ROBERSON, Q. M., COLLINS, C. J. & OREG, S. (2005) The effects of recruitment message specificity on applicant attraction to organizations. *Journal of Business & Psychology*, 19, pp. 319 - 340.
- ROMAN, G.-C., PICCO, G. P. & MURPHY, A. L. (2000) Software Engineering for Mobility: A Roadmap. IN FINKELSTEIN, A. (Ed.) *In Proceedings of 22nd International Conference on Software Engineering Future of Software Engineering*. Limerick, Ireland, ACM Press, pp. 241-258.
- ROWE, G. & WRIGHT, G. (1999) The Delphi technique as a forecasting tool: Issues and analysis. *International Journal of Forecasting*, 15(4), pp. 353 - 375.
- ROYCE, W. (1970) Managing the Development of Large Software Systems. *Proceedings of IEEE Western Electronic Show and Convention (WesCon)* Los Angeles, USA, pp. 382-338.
- SAMPSON, D., KARAGIANNIDIS, C., ANDREA, S. & FABRIZIO, C. (2002) Knowledge-on-Demand in e-Learning and e-Working Settings. *Educational Technology and Society*, Volume 5, No. 2.
- SELF, A. L. & DELOACH, S. A. (2003) Designing and Specifying Mobility within the Multiagent Systems Engineering Methodology. *Special Track on Agents, Interactions, Mobility, and Systems (AIMS) 18th ACM Symposium on Applied Computing (SAC 2003)*. Melbourne, Florida, USA.
- SHAN, L. & ZHU, H. (2004) CAMLE: A Caste-Centric Agent-Oriented Modelling Language and Environment,. *Proceedings of Software Engineering for Large-Scale Multi-agent Systems — SELMAS'04 at 26th International Conference on*

- Software Engineering (ICSE'04), IEEE Computer Society 2004. Edinburgh, UK.
- SIMATIC, M., CRAIPEAU, S., BEUGNARD, A., CHABIDON, S., LEGOUT, M.-C. & GRESSIER, E. (2004) Technical and Usage issues for Multiplayer games. Proceedings International Conference on Computer Games: Artificial Intelligence, Design and Education ( CGAIDE 2004) Reading, UK, pp. 134-138.
- SMITH, D. C., CYPHER, A. & SPOHRER, J. (1994) KidSim: Programming Agents Without a Programming Language. Communications of the ACM, 37, pp. 55-67.
- SOMMERVILLE, I. (2001) Software Engineering Addison Wesley.
- SPELLMAN, P. T., SHERLOCK, G., ZHANG, M. Q., IYER, V. R., ANDERS, K., EISEN, M. B., BROWN, P. O., BOTSTEIN, D. & FUTCHER, B. (1998) Comprehensive identification of cell cycle regulated genes of the yeast *saccharomyces cerevisia* by microarray hybridization. Molecular Biology Cell 9.
- SPENCE, D., CROWCROFT, J., HAND, S. & HARRIS, T. (2005) Location Based Placement of Whole Distributed Systems. In the proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology. Toulouse, France, pp. 124-134.
- STEINKE, S. (1995) Middleware Meets the Network. LAN: The Network Solutions Magazine Volume 10 p. 56.
- STILES, M. (2000) Developing Tacit and Codified Knowledge and Subject Culture within a Virtual Learning Environment International Journal of Electrical Engineering Education 37, pp 13-25.
- STILES, M. & YORKE, J. (2007) Technology supported Learning? Tensions between innovation, and control and organisational and professional cultures. Organisational Transformation and Social Change, Volume 3.
- SUN MICROSYSTEM, I. (2003) System Administration Guide: Naming and Directory Services (DNS, NIS, LDAP). Sun Microsystem, Inc.
- SUTANDIYO, W., CHHETRI, M. B., KRISHNASWAMY, S. & LOKE, S. W. (2004) Experiences with Software Engineering of Mobile Agent Applications. In the proceedings of the Australian Software Engineering Conference (ASWEC 2004). Melbourne, Australia, pp. 339-348.
- SZOLOVITS, P., DOYLE, J., LONG, W. J., KOHANE, I. & PAUKER, S. G. (1994) Guardian Angel: Patient-Centred Health Information Systems. 545 Technology Square, Cambridge, MA, 02139, TR-604, Massachusetts Institute of Technology, Laboratory for Computer Science.
- TAYLOR, R. E. & JUDD, L. L. (1989) Delphi method applied to tourism, New York, Prentice Hall.
- THORN, D., PALMER, I. & WILLIAMS, E. (2004) MMORG on Mobile Devices? Considerations when designing distributed Adventure games. Proceedings

- International Conference on Computer Games: Artificial Intelligence, Design and Education ( CGAIDE 2004) Reading, UK, pp. 150-154.
- TÖRN, A. & ZILINSKAS, A. (1989) Global Optimization. Lecture Notes in Computer Science, No 350, Springer-Verlag, Berlin, p.255.
- TRENCANSKY, I., & CERVENKA, R. (2005) Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS. Informatica (Slovenia), 29, pp. 391–400.
- TSAI, H. K., YANG, J. M. & KAO, C. Y. (2002) Applying genetic algorithms to finding the optimal gene order in displaying the microarray data. In the proceedings of the Genetic and Evolutionary Computation Conference (GECCO), New York, pp 610-617.
- VINOSKI, S. (1997) Integrating Diverse Applications Within Distributed heterogeneous Environment. IEEE Communication Magazine, Volume 14, pp. 46-55.
- VYAS, N. & WOODSIDE, A. G. (1984) An Inductive Model of Industrial Supplier Choice Processes. Journal of Marketing volume 48 pp. 30–45.
- WARE, J. M., WILSON, I. D. & WARE, J. A. (2003) A Knowledge-Based Genetic Algorithm Approach to Cartographic Map Generalisation. Journal of Knowledge Based Systems, Vol. 16/5-6, pp 295-303.
- WAYNE, J. & KARYGIANNIS, T. (1999) Mobile Agent Security. National Institute of Standards and Technology(NIST).
- WEIB, G., FISCHER, F., NICKLES, M. & ROVATSOS, M. (2005) Operational Modelling of Agent Autonomy: Theoretical Aspects and Formal Language. In the proceedings of the Sixth International Workshop (AOSE-2005), AAMAS 2005, pp1-15.
- WEIB, G., FISCHER, F., NICKLES, M. & ROVATSOS, M. (2006) operational Modelling of Agent Autonomy: Theoretical Aspects and Formal Language. Lecture Notes in Computer Science, Springer-Verlag, Berlin, 3950.
- WHITE, J. E. (1994) Telescript Technology. The Foundation of the Electronic Marketplace, General Magic White Paper.
- WHITE, J. E. (1996) Mobile Agents, in Software Agent. IN BRADSHAW, J. (Ed.) In Software Agents,. AAAI Press and MIT Press.
- WILLIAMS, G. & JAHANKHANI, H. (2006) Authenticating E-learners and Virtual Learning systems. IN HOANG NGUYEN, T. & PRESTON, D. S. (Eds.) At the Reader/Probing the Boundary Series. Rodopi Series.
- WILLIAMS, G. B. (2007) Online Business Security Systems, Springer.
- WILSON, I. D., WARE, J. M. & WARE, J. A. (2003) A genetic algorithm approach to cartographic map generalisation. Journal of Computers in Industry Volume 52, pp



291-304.

- WOOD, M. & DELOACH, S. A. (2000) An Overview of the Multiagents Systems Engineering Methodology. In the proceedings of the First International Workshop (AOSE-2000). Berlin, Germany Springer-Verlag, pp.207-221.
- WOOLDRIDGE, M. (1994) Coherent Social Action. In the proceedings of the Eleventh European Conference and Artificial Intelligence (ECAI-94). Amsterdam, The Netherlands, pp. 279-283.
- WOOLDRIDGE, M., JENNINGS, N. & KINNY, D. (2000) The Gaia Methodology for Agent Oriented Analysis and design. Journal of Autonomous Agents and Multi-Agents Systems, Volume 3(3), pp. 285-312.
- WRIGHT, T. (2004) Naming Services in Multi-Agent Systems: A Design for Agent-Based White Pages. In Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS) New York, pp.1478-1479.
- YEO, A., ANANDA, A. & KOH, E. (1993) A Taxonomy of Issues in Name Systems Design and Implementation Communications of ACM SIGOPS Operating Systems Review, vol.27, pp.4-18.
- YIN, R. K. (1994) Case Study Research: Design and Methods Thousand Oaks, CA Sage Publications.
- ZAMBONELLI, F. & JENNINGS, N. R. & Wooldridge, M. (2003) Developing Multiagent Systems: The Gaia Methodology. ACM Transactions on Software Engineering and Methodology, Volume 12, pp. 317–370.
- ZENG, X., MEHDI, Q. H. & GOUGH, N. E. (2004) Implementation of VRML and Java for Story Visualisation Tasks. In the proceedings International Conference on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004) Reading, UK, pp.122-126.
- ZHU, H. (2005) SLABS: A Formal Specification Language for Agent-Based Systems. International Journal of Software Engineering and Knowledge Engineering, Vol. 11, pp. 529-558.
- ZHU, H. & SHAN, L. (2005) Caste-Centric Modelling of Multi-Agent Systems: The CAMLE Modelling Language and Automated Tools. Model-driven Software Development, Research and Practice in Software Engineering Volume II, pp. 57-89.

## **Appendix A - Auto-generated Codes**

### **Auto-generated code for @mobilityRMI function**

```
function mobilityRMIcreatefigure(X1, YMatrix1, yvector1)
%mobilityRMI CREATEFIGURE(X1,YMATRIX1,YVECTOR1)
% X1: vector of x data
% YMATRIX1: matrix of y data
% YVECTOR1: bar yvector

% Auto-generated by MATLAB on 23-Jun-2009 14:21:42

% Create figure
figure1 = figure('PaperSize',[20.98 29.68],'NumberTitle','off',...
    'Name','Genetic Algorithm');

% uicontrol currently does not support code generation, enter 'doc
% uicontrol' for correct input syntax
% In order to generate code for uicontrol, you may use GUIDE. Enter
% 'doc guide' for more information

% uicontrol(...);

% Create subplot
subplot1 = subplot(2,1,1,'Parent',figure1,'Tag','gaplotbestf');
% Uncomment the following line to preserve the X-limits of the axes
% xlim([0 100]);
hold('all');

% Create xlabel
xlabel('Generation','Interpreter','none');

% Create ylabel
ylabel('Fitness value','Interpreter','none');

% Create multiple lines using matrix input to plot
plot1 =
plot(X1,YMatrix1,'Parent',subplot1,'Marker','.', 'LineStyle','none');
set(plot1(1),'Tag','gaplotbestf','DisplayName','Best fitness',...
    'Color',[0 0 0]);
set(plot1(2),'Tag','gaplotmean','Color',[0 0 1],...
    'DisplayName','Mean fitness');

% Create title
title('Best: 1.6362 Mean: 1.6625','Interpreter','none');

% Create subplot
subplot2 = subplot(2,1,2,'Parent',figure1,'Tag','gaplotbestindiv');
% Uncomment the following line to preserve the X-limits of the axes
% xlim([0 21]);
box('on');
hold('all');

% Create xlabel
```

```

xlabel('Number of variables (20)', 'Interpreter', 'none');

% Create ylabel
ylabel('Current best individual', 'Interpreter', 'none');

% Create title
title('Current Best Individual', 'Interpreter', 'none');

% Create bar
bar(yvector1, 'EdgeColor', 'none', 'Tag', 'gaplotbestindiv', 'Parent', subplot2);

% Create legend
legend1 = legend(subplot1, 'show');
set(legend1, 'FontSize', 8);

```

## Auto-generated code for @mobilitysync function

```

function mobilitysyncstoppingoptioncreatefigure(yvector1)
%mobilitysync stopping optionCREATEFIGURE(YVECTOR1)
% YVECTOR1: bar yvector

% Auto-generated by MATLAB on 23-Jun-2009 14:44:58

% Create figure
figure1 = figure('PaperSize', [20.98 29.68], 'NumberTitle', 'off', ...
    'Name', 'Genetic Algorithm');

% uicontrol currently does not support code generation, enter 'doc
% uicontrol' for correct input syntax
% In order to generate code for uicontrol, you may use GUIDE. Enter
% 'doc guide' for more information

% uicontrol(...);

% Create subplot
subplot(1,1,1, 'Parent', figure1, 'Tag', 'gaplotstopping', ...
    'YTickLabel', {'Generation', 'Time', 'Stall (G)', 'Stall (T)'}, ...
    'YTick', [1 2 3 4], ...
    'CLim', [1 2]);
% Uncomment the following line to preserve the X-limits of the axes
% xlim([0 100]);
box('on');
hold('all');

% Create xlabel
xlabel('% of criteria met', 'Interpreter', 'none');

```

```
% Create title
title('Stopping Criteria','Interpreter','none');

% Create bar
bar(yvector1,'Horizontal','on','Tag','gaplotstopping');
```

## Auto-generated code for rastrigin's function

```
function Rastriginsfcncreatefigure(X1, YMatrix1, yvector1)
%Rastrigin's function CREATEFIGURE(X1,YMATRIX1,YVECTOR1)
% X1: vector of x data
% YMATRIX1: matrix of y data
% YVECTOR1: bar yvector

% Auto-generated by MATLAB on 30-Jun-2009 18:13:23

% Create figure
figure1 = figure('PaperSize',[20.98 29.68],'NumberTitle','off',...
    'Name','Genetic Algorithm');

% uicontrol currently does not support code generation, enter 'doc
% uicontrol' for correct input syntax
% In order to generate code for uicontrol, you may use GUIDE. Enter
% 'doc guide' for more information

% uicontrol(...);

% Create subplot
subplot1 = subplot(2,1,1,'Parent',figure1,'Tag','gaplotbestf');
% Uncomment the following line to preserve the X-limits of the axes
% xlim([0 100]);
hold('all');

% Create xlabel
xlabel('Generation','Interpreter','none');

% Create ylabel
ylabel('Fitness value','Interpreter','none');

% Create multiple lines using matrix input to plot
plot1 =
plot(X1,YMatrix1,'Parent',subplot1,'Marker','.', 'LineStyle','none');
set(plot1(1),'Tag','gaplotbestf','DisplayName','Best fitness',...
    'Color',[0 0 0]);
set(plot1(2),'Tag','gaplotmean','Color',[0 0 1],...
    'DisplayName','Mean fitness');

% Create title
title('Best: 7.9274 Mean: 8.7864','Interpreter','none');

% Create subplot
```

```

subplot2 = subplot(2,1,2,'Parent',figure1,'Tag','gaplotbestindiv');
% Uncomment the following line to preserve the X-limits of the axes
% xlim([0 21]);
box('on');
hold('all');

% Create xlabel
xlabel('Number of variables (20)','Interpreter','none');

% Create ylabel
ylabel('Current best individual','Interpreter','none');

% Create title
title('Current Best Individual','Interpreter','none');

% Create bar
bar(yvector1,'EdgeColor','none','Tag','gaplotbestindiv','Parent',subplot2);

% Create legend
legend1 = legend(subplot1,'show');
set(legend1,'FontSize',8);

```

## **Appendix B - Delphi Study**

### **Expert Response 1**

#### **Issue 1**

**Do you follow any particular development lifecycle? (e.g. tradition method of software development, Agile or other).**

We tend to use Agile development practices where we can, especially on small projects. When working on larger projects we are more constrained by ITil and Prince Methodologies

#### **Issue 2**

**What qualities, in order of importance are critical for successful development and implementation of online eHealth? (Qualities such as synchronization, security, concurrency, resilience/persistency, remote messaging, availability or others).**

Security and Patient Confidentiality should always be the main concern for any system used in Health.

Following this the main issues would be resilience/persistency, you need a method of holding the data securely and can survive a power loss.

Losing any data would cause people to lose faith in the system and they would revert back to the older proven methods Synchronization would also be a major issue, I would not allow any system containing to pass patient identifiable information to sync through any network that was not trust authorised, including the 3G network.

#### **Issue 3**

**What are your preferences, if any, for methodology, modelling languages and programming languages?**

I prefer the RAD approach to development, it prevents feature creep, and issues can be resolved much quicker using this approach which leads to higher satisfaction.

For modelling we tend to use UML and use case diagrams and object diagrams, I am also fond of Flow Diagrams to map out how certain events will pan out Programming Languages - C#

#### **Issue 4**

**What development approach do you use (object oriented, agent oriented, Mobile agent oriented or other)?**

OO

### **Issue 5**

**Are there any unanswered questions in the development of for eHealth applications?**

### **Issue 6**

**What can be done to improve existing mobile agent technologies/mobile technologies and methodologies?**

Mobile Technologies have come of age recently, Blackberry, Android and Apple all produce devices that can lever power from multiple areas – GPS tracking to keep staff members safe to integrated database to hold the information, my main concern is battery life.

## Expert Response 2

### **Issue 1**

**Do you follow any particular development lifecycle? (e.g. tradition method of software development, Agile or other).**

Some of the teams I work with follow an agile methodology largely based on scrum with some customizations. The main adaption is the use of Lean workflows and queue limits to try and keep a smooth pipeline of work flowing through.

The other teams I work with use a colloquial version of traditional waterfall.

### **Issue 2**

**What qualities, in order of importance are critical for successful development and implementation of online banking? (Qualities such as synchronization, security, concurrency, resilience/persistency, remote messaging, availability or others).**

Security, availability and reliability are paramount because not only does the online banking site have to do the right things it has to be \*seen\* to do the right things. Reputational damage, fraud and the threat of regulatory sanctions far outweigh other concerns.

Beyond that it is really the same set of qualities as any other ecommerce site - usability, performance (including scalability) and - from the owner's viewpoint - manageability.

**Issue 3**

**What are your preferences, if any, for methodology, modelling languages and programming languages?**

Methodology: agile/lean variant

Modelling languages: very basic UML

Programming languages: java/C-sharp

**Issue 4**

**What development approach do you use (object oriented, agent oriented, mobile agent oriented or other)?**

Largely object-oriented. Mobile variants are not used in my environment and I struggle to see their applicability currently.

**Issue 5**

**Are there any emerging issues in the development of online banking applications?**

This is not my area of speciality though I understand how these intelligent software works. I work mostly on internal bank systems.

**Issue 6**

**What can be done to improve existing mobile agent technologies/mobile technologies and methodologies?**

I have little exposure to mobile agent methodologies or technologies.

Expert Response 3**Issue 1**

**In order of importance how would you rank the following non-functional requirements; synchronisation, remote method invocation, availability and migration, scalability?**

In Chronological order.

1. Availability and migration
2. Scalability
3. remote method invocation
4. synchronisation



**Issue 2**

**Is the methodology phases adaptable for online application such as on line banking, ehealth, gaming and Virtual learning environments? Please see figure 1 and comment?**

Yes, to certain extend. However still I would go through manually to double check.

The reason being the mobility can be made generic to certain extend and highly doubt it can cater all the requirements as application can differ. If it's guaranteed as shown in the diagram then yes it can be.

**Issue 3**

**In term of back end integration, can the development phases be integrated with other systems?**

Yes it can be, as long as the data layer is independent and cross server supportive.

**Issue 4**

**What are the critical functionality issues and challenges in programming for mobility?**

Coding methodology, Coding standard and programming cycle, Language support and Integration.

**Issue 5**

**List 3 main functions of agent software in online banking/ehealth?**

1. Reduces the development time.
2. Cost effective.
3. Reduces the human error.

**Issue 6a**

**How will you test the functions of a mobile agent?**

By performing a small task of conversion process with small known application and checking the functionality of the application on the new platform and running the source application in parallel.

**Issue 6b**

**What data is essential for testing a mobile agent?**

Dependent applications, basic data, both environments and basic knowledge of the test application.

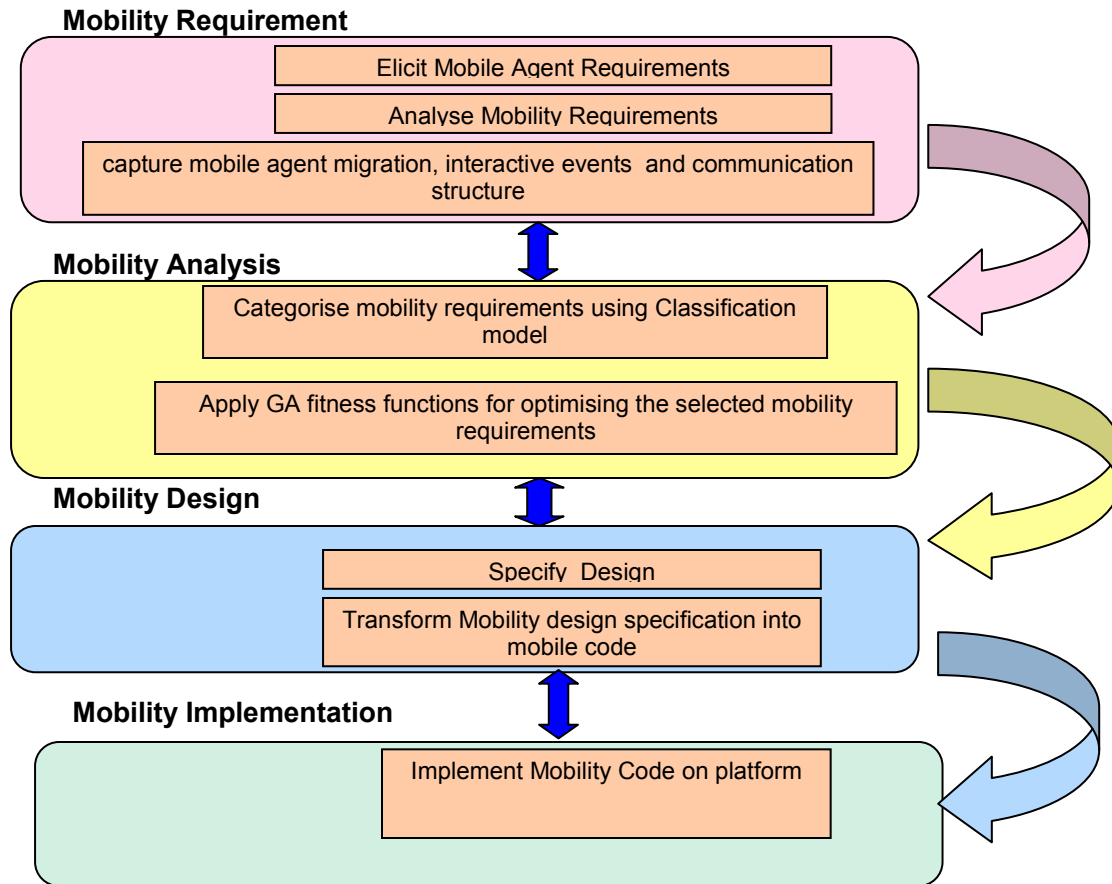


Figure: Mobile Agent Mobility Methodology (MaMM) Phases

## Expert Response 4

### **Issue 1**

**In order of importance how would you rank the following non-functional requirements; synchronisation, remote method invocation, availability and migration, scalability.**

This is application dependent, however from emerging development requirements. The following are essential non functional requirements.

Synchronisation, availability, invocation, migration, scalability.

### **Issue 2**

**Are the methodology phases adaptable for online application such as on line banking, ehealth, gaming and Virtual learning environments? Please see figure 1 and comment?**

Yes. These environments require some aspects of mobility. We suggest that the first phase of the methodology focuses on elicitation of mobility requirements (Delete Agents and Mobile agents). It is also simpler to interpret the “Fitness classification model” phase as the design phase. This means you will not need “design” at the code transformation phase. Rather code the fitness functions and process requirements. These suggestions will streamline the methodology. We suggest you have guidelines for using the methodology.

### **Issue 3**

**In term of back end integration, can the development phases be integrated with other systems?**

To some extent yes and no, as it will be more flexible for systems that deliver business needs that require mobility.

### **Issue 4**

**What are the critical functionality issues and challenges in programming for mobility?**

Synchronisation and language choice. The former is a major challenge for most distributed platforms. The later is a question of language structure. For example java technology copes better with agent based applications. It is also inefficient in some instances, although mobility friendly.

### **Issue 5**

**List 3 main functions of agent software in online banking/ehealth?**

Not a useful question in our opinion, as there are several functions. The obvious ones are remote data access, message sharing with users of the platform.

### **Issue 6a**

**How will you test the functions of a mobile agent? Please explain your answer.**

No immediate view.

### **Issue 6b**

**What data is essential for testing mobility of a mobile agent?**

Non structured data will be useful. This is data from different sources and formats accessible remotely by the agent.

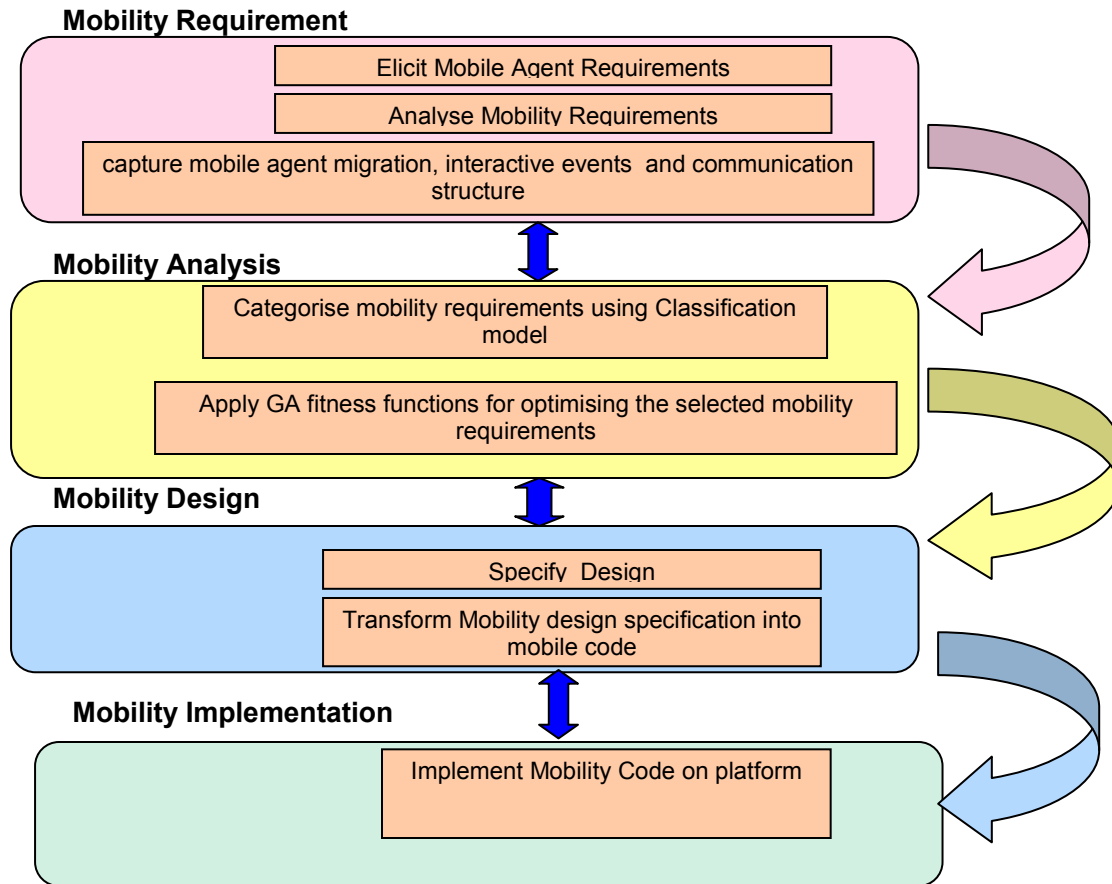


Figure 1: Mobile Agent Mobility Methodology (MaMM) Phases

## **Appendix C - Case study Interviews**

### **Transcript 1**

#### **What do you look out for when developing online banking applications?**

A system that is reliable,

Scalable that is adding components to the system thereby growing the system

User friendly

Maintenance

#### **Do you consider functions like synchronisation and availability?**

Synchronisation is relevant when it comes to database development. You want to have a system that is available so that when something goes wrong or a system fails you can recover.

This is achieved through replication. We replicate data across all the data centres which located at different place.

#### **What about message passing and method invocation?**

We use client /server programming architecture. A thin client will make a call via http to the server. The client will have the URL to launch the application and a communication will be established between the client and the server backend for downloadable java applets.

#### **Do you use agent in your online banking application development?**

No. not really.

#### **What security processes do you have for developing online banking applications?**

We use encryption algorithms for communicating and passing messages between the server and client. We also use SSL

#### **How important is migration in online banking?**

Data migration comes into place when we do implementation from one system to another or from one version to the other. Sometimes third party software is used in which case in house programmers write codes to adapt to the software to do the job.

#### **What processes are involved in ensuring that a data is available?**

This depends on a lot of factors. We recommend that clients have a dedicated digital line or channel that is reliable We use keep record of packet transmitted and availability is ensured through packet headers lost packets are retransmitted, more packets retransmission means there is a problem.

#### **Do you have communication architecture for online banking?**

Yes, it depends on what the application does, what you want to achieve.

We use tunnelling, SSL to ensure secure transmission between the two applications.

**What development process is being used by the development team?**

Agile. Businesses do want application that takes longer time to develop as they will loose business customers and more important the application will be of no use as it will be outdated.

Agile enable codes to push on live environment to be tested and it encounter a problem it will be rewritten push back for testing. agile allows the developer to meet business requirement.

What are the processes involved in implementation?

- Receive codes from developers
- Run out through the UAT environment to test pieces of the code
- Mini data implementation
- Prepare live environment for implementation
- Prepare configuration file
- Prepare database environment
- Prepare back up environment to ensure you can recover in the event of failure or when something goes wrong
- Move to release
- Upgrade different environment
- A test is run in the live environment to check if the system is useable
- Sign off

Thank you for your time.

This expert was contacted again with a draft of the MaMM phases for evaluation. This was he said.

**What is your opinion of the development phases of MaMM on the page? Can you please comment?**

This better methodology than the tradition methodology. Some of the activities involve in migration will slow the process down which means that there will be delay in implementation and inability for the application to be available to the market.

## Interview 2

**Do you use mobile agents in the backend systems development? What role if any does agent play in back end systems in the online banking environment if any?**

Intriguing questions indeed however I want to respond by saying that ;

In the business I am in (Backend Fulfillment systems)we do not use mobile agents while I do see a benefit to using them for online applications but solely for

the purpose of response/scalability within the existing architecture. Im not sure it would be suitable for high volume data intensive processes.

On the backend, we don't use agents at all. Our processes are directed to run on specific server banks and those Banks are load balanced to handle volume. We also incorporate agent listeners based on passed input but again they are not mobile.

## Interview transcript 3

**What functions are important for you when you developing backend application?**

It depends on the bank's usage, what the business community needs it for which is our primary aim as a bank. We also incorporate security standards like encryption. We perform some function in house and some functions are performed offshore.

Robust application. We perform function and non function requirement assessment when we receive business requirement. The business analyst develops technical requirements and gives it the applications development team after which it goes through testing.

**Tell me about performance?**

it depends on the infrastructural back ups.

**What methodology do you use in your applications development?**

Agile. We have different relationship with third parties who does most of the application development offshore which down to some specific application.

## Interview 4

**Tell me about the challenges in designing games in a large organisation such as yours?**

In a large organisation where you're building a platform and building a system, there are lots of people involved.

**How do you go about designing games for more than one person?**

We have to define the multi-player experience, when two or more people play against each other. We wanted people to experience the game as if they were creating their own narrative, as compared to a single player when they're experiencing a story you created for them.

When you listen to a group of people who just played a fun multiplayer game together, it sounds as if they are telling the story of something they just did in reality. Getting to that point in designing a game is challenging.

## Interview 5

### **What functions are important in games design or development?**

The most basic function or element is fun. Games are interactive. They must challenge you, and reward you when you rise to the challenge. In my view, the game begins the moment a person touches a console everything builds from that.

### **Who is a games designer?**

The second skill is being a good communicator because you have to keep communicating with other people on the team. But communication is not only about talking. A good communicator is a good listener above all.

A game designer is at the hub of the development process. He doesn't make the game, but he's the central link to everybody else: the coders, the graphic artists, the sound designers, the scriptwriters and so on.

### **Are there any processes involved in games development?**

I'm giving you a generic answer, because once again, it's on a case by case basis. First you work on the content to outline the main points of the game, key game mechanism, the theme, what we want the player to experience and so on.

Once this is done, the next step is to do a design document. It starts from the concept and elaborates. It defines all game play mechanisms, interface system and so on. It also described the main building blocks of the game.

For example if this was a combat game, we would describe all the fighters, how they looked and what they did. We would also define the art style, which is very important. The goal of this document is to be able to budget the game. Here, you get an idea of what you have to do.

How many 3D designs, how many animations, backgrounds and so on. It is also used by the coders to see what they have to code and where the challenges lie ahead. It is also used to sell the project to a publisher, since most of the time, a publisher will not buy a game on a simple concept.



Anyone can put out a concept, but once again, implementation is the key to success. It reassures them that the developer has apprehended the difficulties and knows how much it's going to cost.

To give you an idea, the first document is usually around 15 pages, the design document is around 150 pages and the third document, called the production document, takes up where the design document left off. You specify everything in detail.

Once again, this is in theory, it doesn't always work like that. In some cases the design document is very small so the publisher can test out ideas, and the production document is built up as we go.

### **What type of programming language do you use in the games programming?**

Good question, You have to learn C++ because it teaches you a way of looking at things.

Already just the notion of Objects.... You have object in the program and objects in the games.

On the other hand, the heaviness of C++ - that it is easily portable and everyone can recover your sources - is really the inverse. If you really want to master your code, you have to learn C. Start with C++, then code in C.

### **How important is iteration and modularity in game design?**

An iterative design is more vital for any product that has to incorporate new or untried features. It will let you fit the design to those deadlines. If development is slipping, you may have to trim some features (or even drop them entirely), but the modularity inherent in the design allows you to use iteration while still keeping control.

### **What other feature(s) indicate a good game design?**

As the game is built, if changes need to be made, the core vision keeps them focused on the final goal.

It ensures that the game features serve a common thematic purpose. For example, if you intend to make a strategy game that assists the player to plan attacks easily, you might think twice about a multilayered interface that, although original, militates against the core vision of ease-of-use.

Most computer games today use high interactivity, as the player has a strongly proactive role.

The story should unfold directly from what the player sees and does, because the player's expectations are that his role is proactive, which means he will be impatient if forced to sit back and be told a story.

Persistence—You can get engrossed for hundreds of hours, experiencing the ultimate in escapist entertainment.

Multiplay—Entertainment software empowers groups of people with the ability to create a mutual narrative.

The inhabitants of the game world can have their own independent existence.

Autonomy—The true payoff of interactivity is that the user can make the product deliver what he wants.

## Interview 6

### **How do you customise a VLE to suit the requirement of your institution?**

VLE could be customised for different solutions or environments. This included upgrading to a later version, adding and customising the user interface, creating graphics, testing and installing extra modules especially some for administration purpose.

### **Tell me about the processes involved from a developer/designer point of view to integrate VLE in the university's environment?**

It is expected that a VLE will be capable of delivering multimedia course materials via a conventional web browser and its associated plug-ins. Other architectures are not excluded provided they offer similar functionality, but they will be unlikely to conform to IMS Content specifications.

our VLE operate in client-server mode and the facilities offered are available from a range of Web browsers on PCs, Apple Macs and Unix-based workstations (including Linux on PCs), although support for offline working may require a client side VLE that can communicate and synchronise with the server based VLE. The server software must run on either Unix or Windows NT Server.

## **Appendix D -Published Papers/ Journal Articles**

1. **Melomey. D.** and Mouratidis, H. (2006). '**Evaluating the Security of Mobile Agent Platforms**'. In Proceedings 1st Annual Conference on Advances in Computing and Technology (ACT'2006) (London, United Kingdom, 24th January), pp. 48 -54.
2. **Melomey. D.,** & Mouratidis, H. and C. Imafidon (2007). '**An Evaluating the Security of Current Approaches for Modelling Mobility of Agent**'. In Proceedings 2nd Annual Conference on Advances in Computing and Technology (ACT'2007), London, United Kingdom, 24th January, pp. 71-78.
3. **Melomey D.,** & C. Imafidon and G. Williams(2007). '**A Comparative Study of Modelling Languages for Agent Systems**'. Systems and Information Science Notes (SISN) volume No. 2 July 2007, pp 207-212.
4. **Melomey D.,** & G. Williams and C. Imafidon (2007). '**Mobility Requirements on Game Platforms: An agent perspective**'. In proceedings of 11th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games. 21st-23rd November , Université de La Rochelle, France (Jul), pp 162-167.
5. **Melomey D.,** & G. Williams , C. Imafidon and R. Perryman (2008). '**A Fitness Function for Capturing Mobile Agent Mobility on Game Platforms**'. In proceedings of 12th International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games. 30th July-2nd August, Galt House Hotel, Louisville, Kentucky, USA.
6. **Melomey D.,** & G. Williams , C. Imafidon and R. Perryman (2008). '**Modelling Mobile Agent Mobility in Virtual Learning Environment (VLE) using Fitness function**'. In proceedings of International Conference on Student Mobility and ICT. 19-20 November 2008, pp 159-165. *University of Maastricht, Maastricht, Netherlands.*

7. **Melomey D.** (2008). **'Mobility Challenges in Online Application Development'**. In 23rd British Colloquium for Theoretical Computer Science. 2008 - BCTCS24 - Durham University, 7-10 April 2008. *Report and abstracts: Bulletin of the EATCS, Number 95, pp262-263*, June 2008, An invited talk.
8. Durkee D., Brant S., Nevin P., Odell A., Williams G, **Melomey D.**, Imafidon C., Perryman R. & Lopes A (2009). **'Implementing e-learning and Web 2.0 innovation: Didactical scenarios and practical implications'**. Industry and Higher Education, Volume 23, Number 4, pp. 293-300.

# Evaluating the Security of Mobile Agent Platforms

Divina Melomey, Haralambos Mouratidis

*Innovative Informatics Group, School of Computing and Technology, University of East London, U.K.*

[dmelomey@yahoo.com](mailto:dmelomey@yahoo.com), [haris@uel.ac.uk](mailto:haris@uel.ac.uk)

**Abstract:** Mobile agents are software entities that can migrate autonomously throughout a network from host to host. This means they are not bounded to the platform they begin execution. This feature of agents makes them a very attractive technology, and in fact it has been argued many times in the literature that mobile agents help to reduce network traffic and perform tasks more efficient. However, security issues have not yet been fully investigated and in fact, mobile agent platforms sometimes they neglect the security issues involved with agent mobility. This paper presents a security related evaluation of 8 main mobile agent platforms.

## 1. Introduction

Developing complex computerised systems has proved to be a difficult task. Actually, it has been argued that developing software for domains like telecommunications represents one of the most complex tasks humans undertake.

Agent technology introduces an alternative approach in developing complex computerised systems. According to this, a complex computerised system is viewed as a multi-agent system in which autonomous software agents (subsystems) interact with each other in order to satisfy their design objectives. Such approach provides designers with more flexibility in their development. The actual design of the system takes place by specifying a multi-agent system as a society, similar to a human society, consisting of entities that possess characteristics similar to humans such as mobility, and intelligence with the capability of communicating.

The concept of a software agent, however, is not uniquely defined. Researchers have given definitions of the concept according to some typical characteristics, some operational

characteristics or some cognitive functions that agents should implement.

One of the most promising features of software agents is mobility. Mobile agents are software entities that can migrate autonomously throughout a network from host to host. This means they are not bounded to the platform they begin execution. However, this feature of agents although makes them a very attractive technology, it also makes the development of platforms (known also as frameworks and environments) that will support mobile agent systems very challenging. One of the main challenges is to develop platforms which will allow a secure migration of mobile agents. Many issues are involved, with respect to security, such as securing the mobile agent from a malicious platform, security the platform from malicious agents and so on.

Although, many different platforms have been proposed by researchers, we believe that security, unlike some other non functional requirements such as performance, has not really thought of during the development of these platforms.

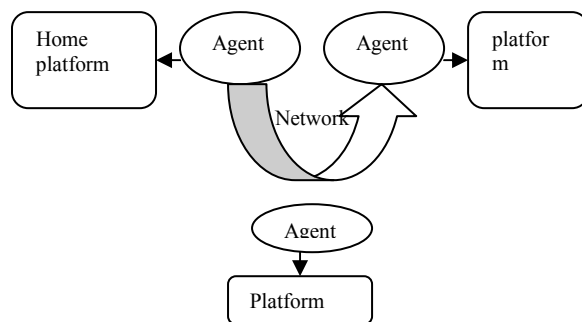
This paper indicates the results of an evaluation, with respect to security, of 8

major agent platforms. Our findings justify the above claim regarding the lack of adequate security mechanisms of these platforms. Section 2 presents a brief introduction to mobile agent migration, whereas Section 3 discusses the security implication of mobile agent systems. Section 4 discusses the evaluation and section 5 concludes the paper and presents ideas for future work.

## 2. MOBILE AGENT MIGRATION

A mobile agent is made up of code and state information, which is needed to perform some form of computation [8]. Therefore, for a mobile agent to execute, an agent platform is required, which is made up of the computational environment.

A mobile agent is characterized by its ability to migrate, during execution, from one host to another as well as between different platforms; even these are running in the same host (see Figure 1 for a partial graphical representation).



**Figure 1: A mobile agent system [8]**

A mobile agent either performs a hop or a multi-hop. A hop is defined as the movement of an agent from its home platform to another platform. Similarly, a mobile agent is said to multi-hop when it hops through various platforms.

## 3. SECURITY IMPLICATIONS IN MOBILE AGENT SYSTEMS

Security threats in mobile agent systems can be categorised into four main categories [8]: (a) Agent to agent attack, when a malicious agent attacks another agent; (b) Agent to platform, when an agent attacks a platform; (c) Platform to an agent, when a platform launches an attack on an agent; (d) External to an agent, when other (non agent) entities attack an agent.

### 3.1 Agent to agent

This is usually in the form of (i) *masquerade*, in which one agent assumes the identity of another to deceive an unsuspecting agent and gain access to sensitive information; (ii) *denial of services to another agent*, which is usually in the form of spam messages sent repeatedly to an agent in order to consume its resources; (iii) *unauthorized access*, where an agent interferes directly with another agent by the invocation of its public methods if the agent's home platform has no control mechanism in place; (iv) *repudiation*, which occurs when an agent denies participation on a transaction; (v) *eavesdropping*, where an agent can gain access to information about other agents' activities, by using services provided by the platform.

### 3.2 Agent to Platform

This is usually in the form of (i) masquerade where an agent tries to gain access on a platform by assuming the identity of another agent; (ii) Denial of Service, in which an agent disallows access to services on the agent's platform; (iii) unauthorized access, in which an agent gains unauthorised access to a platform and is capable of causing harm to that platform.

### 3.3 Platform to an agent

This is usually in the following forms: (i) masquerade, where a platform can assume the identity of another platform in an attempt to deceive another agent with regards to an intended destination as well as its security policy; (ii) denial of service, where a platform ignores service request or may terminate request without notification; (iii) eavesdropping, when confidential and sensitive information is monitored and interpreted by agent platform; (iv) alteration, when an agent arrives at the platform and exposes its code, state and data to the platform. A malicious platform will attempt to modify the code, state and data of the visiting agent unknowingly to the agent. This alters the integrity of the agent.

### 3.4 Other to agent

This occurs in the following ways: (i) masquerade, where an agent makes a request from a platform either remotely or locally. An agent or a remote platform can assume the identity of another to get unauthorized access to resources to which it is not entitled to; (ii) denial of Service, where an entity can access agent platforms server either remotely or locally where an agent with malicious intent can interfere with services that are offered by the platform and inter-platform communication; (iii) unauthorised access ; If remote access to the platform is not properly secured or protected, entities can get access easily and free through scripts available on the internet that can be used to subvert operating system in order to gain control of all systems resources; (iv) Copy and replay; when a mobile agent migrate from one host to the other, it exposes itself to security threat, the message it is migrating with can be intercepted and replay or clone for retransmission [8].

Figure 2 provides a summary of threat per each category.

Threats	Agent to Agent	Agent to Platform	Platform to Agent	Other Entities to Agent
Masquerade	X	X	X	X
Denial of Service	X	X	X	X
Unauthorized access	X	X		X
Eavesdropping	X		X	
Alteration			X	

**Figure 2: Threats per category**

### 3.5 Security requirements

In general, mobile agent systems have the same requirements as general computer systems. These requirements as suggested in [8] are:

1. Confidentiality; any data that is stored privately on a platform or carried by an agent should remain confidential. Intra platform and inter platform communication must also remain confidential and must be ensured by agent framework,
2. Integrity; ensuring that there is no unauthorized modification of the agent framework.
3. Availability; Data and services to both local and remote agents must be made available by the agent platform. Data that is shared must be available in a form that can be used as well as capacity to handle availability of large volumes of request by visiting platform and remote agent.
4. Anonymity; that there should be a balance between the needs of an agent for privacy with the needs of

an agent for platform to hold an agent accountable for their actions.

5. Accountability; all actions must be accountable for by the agent i.e. all processes, operations, meetings of an agent on any given platform. Accountability is necessary for building trust among agent platforms and agent. Audit logs are invaluable source for platform recovery of security breach.

#### 4. Platform Evaluation

Literature provides a wide range of available agent platforms [4, 6, 8, 13]. For the purpose of our evaluation we have identified some of these platforms, which we think are the most appropriate for our research. The selection was based on the following criteria:

- It supports mobility. A basic requirement for any mobile agent infrastructure is its ability to migrate autonomously from one computer or host computer to the other. First, agent should be able to migrate with its entire codes as it goes along and be able to run on any server. Secondly, some servers only require a pre installation of agents' code; such servers do not need transfer of codes to resume execution. Lastly, with some servers, no code is carried by the agent but rather contains a reference to its code base.
- It should be free to use and active. All platforms chosen for this evaluation are available for free download. Moreover, the project is still active meaning the platform is supported either by the developers or from a user group.
- It is written in a language that is widely known with preference to java and scripting language. All the

platforms for this evaluation are written in java except for Telescript which uses the scripting language but it compatible with java platforms and also widely known.

Following the above criteria, we have identified the following platforms for our evaluation: Ajanta, Aglet, Voyager, Concordia, Telescript, Agent Tcl, Tacoma, and JADE.

##### 4.1 Criteria for assessment

Criteria for performing evaluation of the selected platforms have been developed based on the security countermeasures and requirement of mobile agent platforms. In total, forty-one criteria were identified. However, due to lack of space we focus on six of them<sup>1</sup>.

**Criterion 1:** Audit Log for the platform should trace agent falsely repudiating an action.

**Criterion 2:** Safe code interpreter should evaluate all codes.

**Criterion 3:** Agents should be held accountable for their action by using audit trails

**Criterion 4:** The agents function should be encrypted.

**Criterion 5:** support of fault tolerance mechanisms.

**Criterion 6:** Support for authentication and access lists when authorised agents join a transaction

The following table indicates the evaluation of the platforms with relation to the above criteria.

1	2	3	4
NOT SUPPORTED	POORLY SUPPORTED	ADEQUATELY SUPPORTED	FULLY SUPPORTED

<sup>1</sup> Please refer to [15] for the complete list of criteria



CRITERION	PLATFORMS							
	1	2	3	4	5	6	7	8
Criterion 1	1	1	1	1	1	1	1	1
Criterion 2	4	4	1	1	1	1	1	4
Criterion 3	3	1	3	3	1	3	3	3
Criterion 4	4	4	1	2	4	1	1	2
Criterion 5	4	1	4	4	1	4	1	4
Criterion 6	3	1	3	4	4	1	3	4

Key

Platform 1 AJANTA

Platform 2 AGENT TCL

Platform 3 VOYAGER

Platform 4 CONCORDIA

Platform 5 TELESRIPT

Platform 6 TACOMA

Platform 7 AGLETS

Platform 8 JADE

**Table 1: The evaluation**

## 4.2 Discussion about the evaluation

**Criterion 1:** All platforms except Agent Tcl and Telescript provide adequate support. JADE provides full support, mainly because it is based on FIPA specification. Under FIPA 98 specification, an automated mechanism is used to record platform activities in an audit log which is protected. This takes place in order to maintain accountability at platform level, especially with regards to repudiation.

**Criterion 2:** Fully supported by Ajanta, Agent Tcl and JADE. Ajanta provides (or loads) code on demand from a specified agent server. Moreover, agents execute a protected domain that is isolated, in order to prevent agent interference. The function of

the safe code interpreter is to execute commands requiring access to system resources. JADE, Aglet, Voyager and Comcordia use byte code for verification, whereas Agent Tcl uses safe code and Tacoma uses firewalls.

**Criterion 3:** Adequately supported by Ajanta, Voyager, Concordia, Tacoma, Aglet and JADE. Ajanta's full support is based on the fact that the audit trail should indicate the host identity and that of the next (host) as well as its (agent) intended destination. Concordia, Aglets and JADE check if the previous host is a trusted one, whereas Ajanta does this poorly [12].

**Criterion 4:** With encrypted functions, the host must have full control over the mobile code by encrypting it using some agreed conversation algorithms. Ajanta and Telescript fully support this, whereas Concordia provides adequate support.

**Criterion 5:** To avoid tampering and ensure that a code reaches its destination, a Fault Tolerance Mechanism is used. This mechanism when in place helps to achieve replication and voting. Voyager, Tacoma and Concordia fully support this feature, whereas JADE provides adequate support. If an exception is encountered that it cannot be handled, the system's server can take appropriate actions to assist that specific application to recover. Moreover, it should be able to determine the cause of the crash. For this reason, Ajanta supports itinerary abstraction.

**Criterion 6:** Fully supported by Ajanta and adequately supported by Agent Tcl,

Concordia and JADE. JADE achieves this on its runtime environment by enforcing the use of authentication and access lists when joining a transaction. On the other hand, Agent Tcl uses safe Tcl in enforcing access restriction based on its authenticated identity.

The results of the evaluation were analysed graphically and tabulated. Although more than one platforms demonstrated adequate support for most of the evaluation criteria, our analysis of the evaluation demonstrated that JADE offers the best support for security amongst all the platforms, followed closely by Aglets and Agent Tcl. Figure 3 illustrates a comparison of the different platforms against the full set of forty-one criteria.

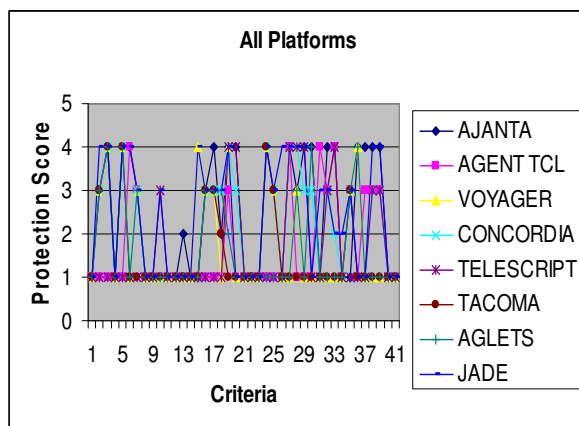


Figure 7: The comparison

## 5. CONCLUSIONS

Security threats to mobile agents have been explored in this paper. A list of evaluation criteria were illustrated together with an evaluation of 8 main mobile agent platforms against those criteria. The presented set of

criteria was derived by considering general security requirement of networked systems as well as special implications of mobile agent systems. The chosen platforms went through the evaluation process and the values assigned were justified on the basis of their ability to meet the requirement in the following order; not supported, poorly supported, adequately supported and fully supported.

Our work is not complete. Future work involves expanding our evaluation criteria to include more specialised criteria, and the development of more experiments in order to validate from an implementation point of view our results.

## 5. References:

- [1] Bellifemine F., Giovanni Caire, Tiziana Trucco, Giovanni Rimassa, "JADE Administrator's Guide", February 2003
- [2] Burbeck K., Garpe D., Nadjm-Tehrani S., "Scale-up and Performance Studies of Three Agent Platforms" Proceedings of International Performance, Communication and Computing Conference, Middleware Performance workshop, (Phoenix, Arizona, USA), pp.857-863, Apr. 2004
- [3] Dancus Andrei, "JADE- A FIPA compliant Java Agent Development Framework" Worcester Polytechnic Institute, Spring 2002
- [4] Ferguson, Tracy (2004), "Mobile Agents and E-Commerce", CPSC 820 Final Report, Clemson University, Computer Science Department, April 2004
- [5] FIPA Agent Management Specification, 2004/18/03
- [6] Fischmeister S., Vigna G. & Kemmerer R.A. "Evaluating the Security Of Three Java-Based Mobile Agent Systems" Proceedings of the International Conference on Mobile Agents (MA

2001) 31-41 LNCS 2240, Springer-Verlag Atlanta, GA, December 2001

[7] Helmer Guy, Johnny S.K. Wong, Vasant Honavar, Les Miller, Yanxin Wang. "Lights agent for Intrusion Detection" The Journal of Systems and Software 67, pp 109-122, 2003

[8] Jansen, Wayne & Karygiannis, Tom "Mobile Agent Security, National Institute of Standards and Technology (NIST) special publication 800-19, October, 1999

[9] Kadhi N., Boury P., "Statistic Analysis of Java Cryptography Applets". In proceeding of ECOOP2001 (Budapest) Workshop on java Formal Verification, 2001

[10] Kapse, Padma " Security in Mobile Agents", CSE Security Research Group or Secure Systems Research Group, 2003

[11] Karnik N, Vora M, Ahmed T., & Singh R. "Mobile Agent Programming in Ajanta" Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, Austin, Texas, May 1999, pp. 190-197

[12] Karnik N and Tripathi "design Issues in Mobile Programming Systems 2, IEEE Concurrency 1092-3063, 1998

[13] Shiao, Dan (2004), "Mobile Agent: New Model of Intelligent Distributed Computing ", IBM, China, October 2004.

[14] Wheeler Thomas, "Voyager Architecture Best Practices" Recursion software Inc, March 2005

[15] Melomey, Divina, "Security of Mobile Agent Platforms", MSc Thesis, University of East London, 2005

.

# An Evaluation of Current Approaches for Modelling Mobility of Agents

**Divina Melomey, Haralambos Mouratidis, Chris Imafidon**

Innovative Informatics Group

School of Computing and Technology, University of East London, U.K.

{divina, haris , chris12}@uel.ac.uk

## **Abstract**

The development of agent-based systems requires methodologies and modelling languages that are based on agent related concepts. Towards this direction, research has proposed a large number of Agent Oriented Software Engineering (AOSE) approaches to modelling mobility of agents. This paper will evaluate the current approaches and methodologies with respect to modelling mobile agent systems and it will propose a number of concepts required to adequately model agent mobility.

## **1. Introduction.**

An agent is a computer program that demonstrates characteristics such as social ability, reactivity, pro-activeness, and autonomy (Wooldridge and Jennings 1995). Mobile agents are special types of agents that possess all the characteristics of an agent but they also demonstrate the ability to move or migrate from one node of a network to another. Mobile agents (Milojicic et al., 1999) (Jansen and Karygianni, 1999) have received considerable attention from industry and research community, since their special characteristics help to address network issues such as network overload, network latency, and protocol encapsulation, just to name a few.

Due to the popularity of the agent technology, mainly in the research environment, there has been an influx of software engineering methodologies for the development of multi-agent systems (i.e. systems that consist of more than one agent). Current approaches model static agents and little or no attention has been given to the modelling of mobile agents. Nevertheless, for mobile agent systems to become widely acceptable there is a need for a methodology

to be developed which addresses various issues related to the mobility of agents. For instance, methodologies should assist developers to determine at the onset which agents should remain stationary and which needs to migrate on the network and hence how these could be modelled.

This paper provides an overview of current approaches and modelling languages for modelling multi-agent systems, and their limitations with respect to mobile agent systems modelling. It proposes a set of concepts and a modelling language necessary for modelling mobile agent systems.

The layout of the paper is as follows; section 1 provides an introduction to agent technology while section 2 presents the state of the art and limitations (with respect to agent mobility) of existing approaches and modelling languages. Section 3 presents the concepts to model mobility of mobile agents while section 5 concludes the paper and also presents future works.

## **2. State of the art and Limitations of existing approaches and modelling languages:**

A number of approaches and modelling languages have evolved since the emergence of agent technology. Notably among approaches for modelling agent systems that have emerged are Gaia (Wooldridge et al., 2003), MESSAGE (Caire et al., 2000), TROPOS (Bresciani et al., 2003) and Multi Agent Systems Engineering (MASE) (Self & DeLoach, 2003), Prometheus (Padgham and Winkoff, 2002). Some of the approaches mentioned above mostly concentrated on design issues such as modelling static mobility. Few attempts were also made at modelling the dynamics of mobility of the agents. There are inadequate concepts to specifically model mobility of mobile agents.

Chhetri et al. (2006) developed ontology that describe concepts, and the relationships that exist between them to model mobility issues. The core concepts defined does not include a continuous link that depicts mobility among different components or interactive agents, which presumes the survival of mobile agents. Their ontology did not implicitly define what agent and mobile agent are but presume an agent becomes a mobile agent when it is assigned a role and also see mobility as attribute. This therefore implies that a designer cannot reason about mobility during the requirement phase of systems development. Their ontology did not specify security to mobile agent.

## **2.1 Overview of Current Approaches**

There are other approaches to model mobility of agent. These approaches do not form a complete methodology on their own, but they stem from the component, elements and diagrams of the Unified Modelling Language (UML). UML provides unification and formalization for methods of numerous

approaches to the object oriented software systems lifecycle while Agent UML provide same functionality but for agent oriented systems. An approach such as Gaia was not built on UML. Agent Modeling Language (AML) is also another modelling language specified as an extension to UML 2.0 (Cevenka et al., 2005) (Cevenka et al., 2005b) (Cevenka and Trencansky, 2004). Some approaches such as Gaia did not use UML at all.

## **2.1 Gaia Methodology**

The Gaia methodology (Juan et al., 2002) focuses on analysis and design of agent based system. It provides analyst tools to develop a system from the systems requirement to detailed design which allows for direct implementation of the system (Wooldridge et al., 2000). Gaia models a complex system using agent concepts. Gaia defines responsibility when it assigns roles to agents.

However, Gaia lacks concepts and graphical notations to support modelling and reasoning about mobility of agents' vis-à-vis their social interaction with each other in a multi agent environment.

## **2.2 TROPOS**

TROPOS as a requirements-driven methodology was developed to support analysis and design activities (Bresciani et al., 2004) (Castro et al., 2002). TROPOS covers the early and late requirement phases, as well as the architectural design and implementation phases. Its greater strength lies only in identifying early requirements for the system in spite of the fact that it has a broader coverage of the entire software development process.

However, TROPOS has not been developed with mobile agents in mind and therefore it fails to provide the necessary processes and concepts to model mobility of agents (Bresciani et al., 2004).

### **2.3 MaSE Methodology**

Multi-agent Systems Engineering (MaSE) is a methodology (DeLoach & Self, 2001), (DeLoach, 2004), (DeLoach, 2006). From all the available AOSE methodologies, it is only MaSE that managed to model some aspects of agents' mobility using UML. In particular, MaSE makes provision of tools which enables developers/designers to specify where and which location an agent can migrate to, which task and communication processes should be retained and which should not (Self & DeLoach, 2003). However, they only focused on the output models of the analysis phase of the systems lifecycle, and they also fail to identify why mobility is needed by some agents, and the association with the system requirements.

### **2.4 AUML Extensions**

As mentioned above, apart from the methodologies for the development of agent systems, there have been few efforts to develop modelling languages and definition of some concepts that can be employed for the modelling of agent and mobile agent systems. In particular, Poggi et al. (2004) extended AUML deployment and activity diagrams with concepts and notations such as home, mobility path, destination, visitor, dotted lines to represent messages and dash lines with arrows pointing towards platforms that a mobile agent might be visiting. These concepts and notations have been introduced to extend the deployment diagrams (Poggi et al. 2004). All these concepts and notations introduced are geared towards modelling the

static movement of the mobile agent, without paying particular attention to the dynamic mobility of an agent. Another important issue for mobile agent systems is security. However, the proposed concepts and notations fail to allow developers to consider security issues that might be present on their mobile agent systems.

Furthermore the issue of time was also not addressed by the proposal put forward by (Poggi et al., 2004). For instance, it is not possible to model when a mobile agent decides to move from one node to another. .

Regarding activity diagrams, Poggi et al. (2004) introduced concepts such as return path, bounced failure and notations to indicate two statements with two arguments. These concepts and notations are intended to capture the dynamics of the agents i.e. concurrency, sequence and iterations of the movement of the mobile agent. This extension only captured the sequence of activities and knowledge provided by the designer so that a mobile agent can make an informed choice.

However, this was not fully realized since there was no continuous established link for which the mobile agent can make independent decision on its movement i.e. to and from its previous platform. There was also no indication whether a mobile agent has the necessary permissions to visit certain platforms. In addition, there was no mention or indication whether the agent has any kind of itinerary or not, and the kind of activities it does on its way to accomplish a task or a goal.

Similar to the work by Poggi et al (2004), Baumeister et al (2003) presented new stereotypes such as mobile, mobile location, at location, clone and move to model mobility in mobile systems which is an

extension to activity diagram. New concepts such as mobile objects, locations and actions to moving mobile objects were introduced by the authors. Location that is contained in another is called nested location was also considered. Two notional variants were also introduced. These are location and responsibility centred. These provide answers such as who is performing an action and where the action is being performed. Swimlanes were introduced to represent objects showing who is performing a particular action as well as mobility of an object with respect to topology of location

However, the concept of nested location was not properly defined and illustrated. The idea of mobile location lacked clarity. Even though the extension to the activity diagram was to model mobility in mobile systems, concepts introduced has no direct bearing to neither agents nor mobile agents. All references were made to objects.

(Kosiuczenko, 2003) introduced the stereotype class *move* in sequence diagram to model mobile objects. It further introduced stereotypes for cloning objects which are *create* and *copy*. Mobile objects in this extension can change its location when it performs a jump action. Concepts on nested topology were also presented by the authors. Changes do take place during the life line of a mobile objects and hence the ability to trace mobile objects that perform the jump action. The lifeline therefore contains all the jumps right from the first place the mobile objects appeared. According to the author, the lifelines contain all jump arrows of the mobile object and its host and ends where the lifeline of the mobile object ends or terminates.

This extension, however, focused on objects and not agents. There is also no formal

semantics for modelling the sequence diagram, hence lack of tool support to aid the analyst to perform a thorough analysis of systems.

## 2.5 Agent Modeling Language (AML)

AML is specified as an extension to UML 2.0 is a semi visual modelling language. It is used to specify, model and document systems that incorporate concepts and features of multi agent systems theories and existing abstract models such as TROPOS, Gaia, MESSAGE, UML, PASSI, Prometheus and MaSE (Trencansky and Cervenka, 2004b). In modelling the deployment of Multi Agent systems (MAS), AML attempted to provide support for mobility by identifying the following main elements: the agent execution environment, the hosting property, dependencies i.e. the move and clone, and lastly actions of move and clone (Trencansky and Cervenka, 2004b). However, there was no supporting model or construct to model the mobility of the agent. No mention was made of mobile agents and how their movement can be captured. Clearly, AML focus is not on mobile agent but rather on multi agent systems.

## 3. Building an Ontology for Modelling Agent Mobility

As mentioned and proved above, there is no single approach to guide the designer to reason about mobility from conception of an idea to its completion. An approach for modelling mobility issues of agent-based systems should have a set of modelling tool, a highly expressive modelling language and well documented semantics to assist software engineers to reason and model agent mobility issues as well as incorporating security where necessary.

Below we present a list of concepts (along with their definition) that we have found are necessary to be included in a complete ontology for modelling mobile agent systems.

### 3.1 Mobility Concepts

To overcome some of the limitations identified in the earlier section, this paper therefore presents a new and enhanced set of concepts to model the mobility of agents. Due to lack of space we present only brief definitions of concepts. These concepts are software agent, stationary agent, mobile agent, platform, home platform, host platform, , summit, mobility link, weak mobility, strong mobility, itinerary, task, goal, zone, permissions, sleep mode and knowledge base.

#### **Software Agent**

As mentioned above, software agent can be either stationary or mobile. It is important therefore to allow developers to model both types of agents. An agent comprises of code and state information needed to carry out some kind of computation. We differentiate a software agent to stationary agent and mobile agent.

**Stationary Agent** is an agent that is stationary. In other words, an agent which executes in the place it started. Stationary agent does not move.

#### **Mobile Agent**

This is an agent capable of moving among different platforms.

#### **Platform**

For an agent (and therefore mobile agent) to run, a platform is required; in other words an agent platform provides the computational environment in which an agent operates. For

the purpose of modelling mobile agents, our work models a platform as networks of computers or independent nodes, irrespective of size. A platform offers resource services to other agents that enter it. For modelling mobility, two types of platforms are required.

#### **Home Platform**

This is the location where an agent originates.

#### **Host Platform**

Any platform a mobile agent migrates to apart from its home platform.

#### **Summit**

Summit allows two or more agents and/or mobile agent to meet in the same computer. Here, a mobile agent decides to migrate to meet with another stationary agent on a server platform for a service.

#### **Mobility Link**

A Mobility Link establishes a link or a session between or among agents. A link can be established only if the agents can identify each other. A mobility link can be terminated by either agent at both ends of the established mobility link. While a link is established, an agent must not move to another place or location on the platform; should this happen, the mobility link will be implicitly terminated. Therefore in this context mobility link will be used to synchronise agents that want to meet for a summit. Mobility link allows a connection to be made regardless of the distance. It also enables a mobile agent to obtain a service remotely and the return to its home platform. A user's agent for example should be able to obtain flight information and book a flight for the user. On its return to the home platform, the user's mobile agent should be



able explain to the user, the type of ticket booked, be it first class or economy.

### **Weak Mobility**

This involves a situation where an agent gathers or stores no information on previous host visited. This is suitable to collect on line data to perform simple control and configuration tasks from several networks elements. It also leads to the reduction of network load.

Weak mobility copy only code. Program execution starts from initial state e.g. java applets

### **Strong Mobility**

This preserves accumulated information upon migration. In addition it is able to process data from network elements. It is also able to preserves its state and form during previous visits.

Strong mobility copies code and execution. It resumes execution where it stopped but doesn't necessarily have same resources on current platforms.

Migration process ceases at originating site.

### **Itinerary**

Itinerary represents the mobility plan of the mobile agents' movement.

### **Task**

A task is any action or series of actions an agent or mobile agent can perform as part of its itinerary and its goals.

### **Goal**

A goal is a specific objective an agent aims to accomplish. This is what motivates it to meet for a summit, hence establishes a mobility link in order to achieve this goal.

### **Zone**

This a collection or a group of platforms operated by the same authority. To this end,

a source mobile agent should provide enough proof to the destination zone else access will be denied.

A mechanism therefore will be provided to verify the authority of a mobile agent migrating from zone to zone.

Hence authority will limits what platforms and agents can do at any point in time.

### **Permissions**

Permissions will grant the right to execute an instruction or perform an action i.e. ability to create another agent and to grant them rights to use certain resources and a life to live such as a few hours or days after which it terminates.

### **Sleep Mode**

This affects and monitors changing conditions. This occurs the moment a mobile agent put itself to sleep until such as a time it needs to be active. For example when a trip is book for a later date, on the day of the flight, the mobile agent awake and inform about any delay and/or of the details of the trip.

### **Knowledge Base**

These are rules that will be loaded in to the mobile agent at the start time which will enable the mobile agent to make an informed decision.

## **4. Conclusion and future work**

In this work, we have examined the existing methodologies and approaches used in modelling agent mobility; we have presented critical concepts needed to model mobility. There are still more of these mobility concepts than space will allow us. Our primary aim, in this paper, was to evaluate all current approaches for modelling mobility. Our research indicated lack of a complete approach to model all the

issues related to modelling mobile agents. The approaches are also not complete in themselves, in that they lacked proper illustrative examples; all examples used are not complex enough to reveal weaknesses in the approach.

In our future work, these concepts will be modelled and evaluated using an exemplar with a supporting modelling tool as well as a supporting documentation.

## 5. References:

- Bauer B., Müller J. P., Odell J. "Agent UML: A Formalism for Specifying Multiagent Interaction," 22nd International Conference on Software Engineering (ISCE), *Agent-Oriented Software Engineering*, Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pp. 91-103, 2001.
- Baumeister, Nora Koch, Piotr Kosiuczenko, and Martin Wirsing. "Extending Activity Diagrams to Model Mobile Systems". In M. Aksit, M. Mezini, and R. Unland, editors, *Objects, Components, Architectures, Services, and Applications for a Networked World*. International Conference NetObjectDays, NODe 2002, Erfurt, Germany, Oct. 7-10, 2002. Revised Papers, volume 2591 of LNCS, pages 278-293., 2003.
- Bergenti, Federico; Gleizes, Marie-Pierre; Zambonelli, Franco (Eds.) Kluwer Academic Publishing (available via Springer), 2004.
- Bresciani, P. Giogini, F. Grunchiglia, J. Mylopoulos, and Perini A. "Tropos: An Agent-Oriented Software Development Methodology". *Journal of Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers, 2004
- Caire, G., Coulier, W., Garijo, F., Gomez-Sanz, J., Pavon., J. Kearney, P. and Massonet. P." MESSAGE: A Methodology for the Development of Agent-Based Applications, To appear at *Methodologies and Software Engineering for Agent Systems*, edited by Federico Bergenti, Marie-Pierre Gleizes and Franco Zambonelli, to be Published by Kluwer Academic Publishing, 2004
- Castro, J., Kolp M., and Mylopoulos. J. "Towards Requirements-Driven Information Systems Engineering: The Tropos Project". In *Information Systems*, Elsevier, Amsterdam, The Netherlands, 2002.
- Cervenka R. and Trenansky I. "Agent Modeling Language". Version 0.9. Technical report, Whitestein Technologies, 2004.
- Cervenka R. and Trenansky I., and Calisti M.. "Modeling Social Aspects of Multiagent Systems. The AML Approach". In J.P. Muller and F. Zambonelli, editors, *The Fourth International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 05). Workshop 7 : Agent-Oriented Software Engineering (AOSE)*, pages 85-96, Universiteit Utrecht, The Netherlands, 2005
- Cervenka R. and Trenansky I, and Calisti M., Greenwood D.. "AML: Agent Modeling Language. Toward Industry-Grade Agent-Based Modeling". In J. Odell, P. Giogini, and J.P. Muller, editors, *Agent-Oriented Software Engineering V: 5<sup>th</sup> International Workshop, AOSE 2004*, pages 31-46, Springer-Verlag, Berlin, 2005.
- Cysneiros L. M., Werneck V. and Yu E. "Evaluating Methodologies: A Requirements Engineering Approach Through the Use of an Exemplar". *The Journal of Computer Science and Technology*, Vol. 5No.2
- DeLoach Scott A."Multiagent Systems Engineering of Organization-based

Multiagent Systems". 4th International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'05). May 15-16, 2005, St. Louis, MO. Springer LNCS Vol 3914, Apr 2006, pp 109 - 125.

DeLoach Scott A., Wood Mark F. and Sparkman Clint H., "Multiagent Systems Engineering", The International Journal of Software Engineering and Knowledge Engineering, Volume 11 no. 3, June 2001.

DeLoach Scott A.. "The MaSE Methodology. In Methodologies and Software Engineering for Agent Systems". The Agent-Oriented Software Engineering Handbook Series: Multiagent Systems, Artificial Societies, and Simulated Organizations, Vol. 11. Bergenti, Ivan Trencansky, Radovan Cervenka: Agent Modeling Language: A Comprehensive Approach to Modeling MAS. Informatica (Slovenia) 29(2) 391-400(2005)  
Jansen,W. and Karygianni,T.(1999)"Mobile Agent Security, National Institute of Standards and Technology (NIST) special publication 800-19 , October,1999

Jennings N. R., Sycara K. and Wooldridge M. (1998) "A Roadmap of Agent Research and Development" International Journal of Autonomous Agents and Multi-Agent Systems 1 (1) 7-38.

Jennings N. R, Wooldridge M."Agent-Oriented Software Engineering (2000)". Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99)  
Juan T.,Pearce A., and Sterling L.,. "ROADMAP: Extending the Gaia Methodology for Complex Open Systems". In Proceedings of the first international joint

conference on Autonomous agents and multiagent systems (AAMAS2002), Bologna, Italy, pages 3--10, 2002.

Kang M, Taguchi K." Modelling Mobile Agent Applications by Extended UML Activity Diagram". ICEIS(4) 2004: 519-522  
Kosiuczenko P.."Sequence Digrams for Mobility". Krogstie J. (ed.): Advanced Conceptual Modeling Techniques: ER 2002 Workshops, ECDM, MobIMod, IWCMQ, and eCOMO, Tampere, Finland, October 7-11, 2002, LNCS 2784, Springer, Berlin, 12 pages, 2003.

Milojicic D., Kotz D., Lange D., Petrie C., Rygaard C. "Mobile agent applications" IEEE Concurrency July to September 1999

Szolovits P., Doyle J., Long W. J., Kohane I and. Pauker S. G. "Guardian Angel: Patient-Centred Health Information Systems". TR-604, Massachusetts Institute of Technology, Laboratory for Computer Science, 545 Technology Square, Cambridge, MA, 02139, May 1994

White J.E. "Mobile Agents, in Software Agent", JM Bradshaw, Editor. MIT Press ... In Software Agents, J. Bradshaw, editor, MIT Press, 1996, pp. 437-472

Wooldridge, M., Jennings, N.R. and Kinny, D. "The Gaia Methodology for agent oriented analysis and design". Autonomous Agents and Multi-Agent Systems, 3(3), 2000, pp 285-312

Yu, E., Cysneiros L.M. "Agent-Oriented Methodologies- Towards a Challenge Exemplar" in Proc of the 4<sup>th</sup> International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS 2002) Toronto May 2002.

Padgham L. and Winkoff, M., Prometheus: A Methodology for Developing Intelligent Agents , Proceedings of the Third International Workshop on Agent-Oriented Software Engineering, at AAMAS 2002, July,2002, Bologna, Italy.

Poggi A., Rimassa G., Turci P., Odell J., Mouraidis H., and Manson G. Modelling Deployment and Mobility Issues in Multiagent Systems using AUML, in Agent Oriented Software Engineering IV, P. Giorgini, J. P. Muller, J. Odell (eds.), *Lecture Notes in Computer Science 2935*, Springer-Verlag 2004.

Self A. & DeLoach Scott A. "Designing and Specifying Mobility within the Multiagent Systems Engineering Methodology". Special Track on Agents, Interactions, Mobility, and Systems (AIMS) at The 18th ACM Symposium on Applied Computing (SAC 2003). March 9 - 12, 2003, Melbourne, Florida, USA.

Wooldridge, M. and Jennings N. R. (1995), "Agent Theories, Architectures, and Languages: a Survey," in Wooldridge and Jennings Eds., *Intelligent Agents*, Berlin: Springer-Verlag, 1-22

Zambonelli F., Jennings N.R and Wooldridge M. "Developing Multiagent Systems: The Gaia Methodology". *ACM Transactions on Software Engineering and Methodology*, 12(3): 317-370, July 2003.

# A Comparative Study of Modelling Languages for Agent Systems

Divina Melomey, Chris Imafidon and Godfried Williams

Email: {divina; chris12; G.Williams}@uel.ac.uk

School of Computing and Technology

University of East London, Dockland Campus

London, UK

## Abstract:

Agent Oriented Software Engineering (AOSE) is an emerging field in Software Engineering. This paradigm is based on the concept of agent, an autonomous computing entity. Some of the benefits AOSE provides to system developers are concepts and notations that relate to real life situations. These concepts include knowledge, behaviour, beliefs and desires, as well as characteristics similar to human's intelligence and mobility. This paper examines recent research in agent modelling languages and compares common modelling languages such as Agent Unified Modelling Language (AUML), Specification Language for Agent-Based Systems (SLABS), A Caste-Centric Agent Modelling Language and Environment (CAMLE), Agent Modelling Language (AML) and Autonomy Specification Language (ASL) for modelling an agent with respect to mobility. The criterion for comparison is based on functions of modelling languages, characteristics as well as semantic structure.

**Keywords:** agents, mobile agent, modelling languages.

## 1. Introduction

A modelling language is a form of communication tool that enhance communication between software developers and management. Modelling languages guides developers to clearly represent internal and external structures for textual and visual representation. Software modelling languages enable software developers to specify requirements of software systems during software development by conceptualizing the world in the form of entities known as agents. Software agents are characterised by autonomy, social ability, proactively and reactivity. These characteristics of the agent, calls for a more robust modelling language in capturing agent requirements. These characteristics dictate the need for an agent modelling language capable of capturing these requirements.

Software agents draw its fundamental concepts historically from artificial intelligence, distributed computing and objected oriented systems engineering.

There is the need to advance pre existing modelling languages to model the evolving requirements of agents. Most of these modelling languages have been used to express knowledge in domain areas with respect to goals, tasks and vocabulary for expressing concepts underlying agent applications.

Section 2 presents motivations underpinning the need to study agent based modelling languages. Section 3 gives an overview of existing modelling languages. Section 4 highlights criteria for assesses the effectiveness of modelling languages. Section 5 presents the results on the comparison; while section 6 presents conclusions and future work.

## 2. Necessity for Modelling Languages

Unlike object oriented systems development methodology, AOSE [12] has not reached its maturity stage where issues such as modelling languages for requirements specification phase through to the implementation of the entire software development process is captured. It is paramount therefore to have modelling language(s) that models interaction of agents, their behaviours from the requirement phase throughout to implementation. Modelling languages provide a vivid description of agent systems and also serve as tools for capturing the reasoning underlying mobility. Issues that usually arise when modelling agent systems include agent representation, validation, verification and representation of mathematical and linguistical requirements. These issues have not been addressed and seem to have been ignored.

### 2.1 Common Requirements

The main requirements that need to be captured comprise functional and non functional depending on the area of application. Agent behaviour, service and application needs, stake holders, users as well as their interaction with the proposed system are considered as functional requirements. Non functional requirements cut across issues such as mobility, security, performance, and synchronization and user friendliness of the agent. This paper discusses mobility requirements of an agent and the elements necessary for developing a methodology that could capture such requirements.

### 2.2 Challenges Associated with Mobility

Modelling languages are required to capture both internal and external structures of the agent. Even though it is portrayed that agents must have control over their internal structures, there is the need to show the transition from one phase to the other as well as the point which control is left entirely to the agent.

The modelling of agent systems requires a combination of visual and formal languages. Formal specification tends to provide solutions that address weaknesses associated with

visualization Formal specification enables models to be defined using precise semantics. Furthermore, it facilitates the transformation from one phase to another, for example from the analysis phase to systems design phase of the development process. This therefore requires some specialist skills on the part of developers. This is effective for communication amongst developers but ineffective and inappropriate for communication and discussion with stakeholders. Formalising visual languages for conceptual modelling comes with a set of challenges such as, unambiguities in meaning and expression of graphical notations. In the next section, we will examine some of the existing modelling languages and their effectiveness in the various phases of the development process.

### 3. Overview of Modelling Languages

There are quite a number of modelling languages for modelling mobile agents and agent systems, most of which draw concepts from unified modelling languages [7]. This paper presents both text formalisation and visual based languages.

A visual language allows the domain knowledge developers to assemble programs quickly from existing components with it related operations. Visual language offers an added advantage when there is a match between the system to be modelled and the visual abstract. On the other hand, human skills present a higher skill level in terms of knowledge using textual languages with its associated tool support.

The main languages assessed are Agent Unified Modelling Language (AUML)[6], Specification language for Agent-Based Systems (SLABS)[4], A Caste-Centric Agent Modelling Language and Environment(CAMLE)[9], Agent Modelling Language (AML)[13] and Autonomy Specification Language(ASL)[1].

#### 3.1 AUML

AUML is an extension to the unified modelling language. There are no restrictions to the extensions one can make to UML. Some researchers have made attempts on extending UML. Mouratidis et al. [2] provided extensions on deployment and activity diagrams to model agent mobility. Similarly, another approach to model mobility was the extension of activity diagrams using UML 1.5 [11].

#### 3.2 AML

AML has features for capturing multi agent systems. AML combines both visual and formal language for modelling and agent specification. It draws its concepts from multi agent systems theory. AML also specifies models and document systems by using the extending UML 2.0.

#### 3.3 SLABS

SLABS provide the developers with language facilities together with features for formal specification as well as the verification of agent based systems. Its focus geared toward the development of scale complex system. SLABS is based on a generalised model of agents rather than a specific agent

theory and it is decomposable. It integrates new concepts such as caste and provides language facilities AOSE.

#### 3.4 CAMLE

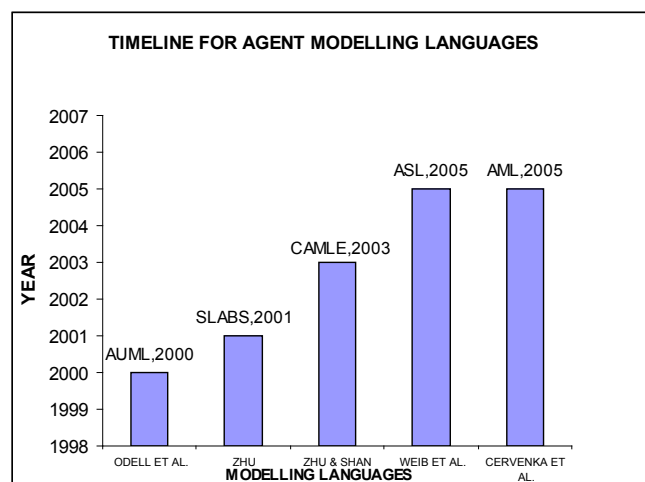
CAMLE is a language based on the notion of caste and draws on the concepts of SLABS. Caste by definition is a set of agents with the same behaviour and structure. SLABS combine both graphical modelling with formal specification language by automation. CAMLE introduced visual models at the design stage of the development process. These models are caste, collaboration and behavioural. Diagrams in caste model specify caste in the systems and their relationships including their movement from one caste to the other. Collaboration model includes diagrams organized in a hierarchical order depicting the interaction of agents and their relationship in the system. Lastly, the behavioural model diagrams define how agents decide on what action to take and how it changes its states depending on a given scenario. All these models come with its associated notations well defined.

#### 3.5 ASL

ASL has its strength in the operational modelling for specifying the autonomy of the agent. An ASL concept defines roles through a set of activities as well as specifying the behaviours that conforms to or deviates from accepted norms of agents system. In specifying the behaviours it enables behavioural prediction of agent through the roles they assume. Furthermore, ASL enables software designers to specify autonomy of agent as well as allowing the detection and resolution of induced conflicts that occur during runtime.

The modelling languages mentioned in this section have each their own strengths and weaknesses. There is therefore the need for cohesion of models and concept through each phase of the software development process. Below is a timeline of modelling languages used to model agent systems and mobile agent since year 2000.

Figure1. Agent Languages Timeline.



Criteria have therefore been defined based on the function and attributes of modelling languages for the comparative analysis. These functions are further dependent on the type of

application for which mobile agent and for that matter agent has been cited as a possible solution.

#### 4. Criteria for Comparative Analysis

Criteria for comparison have mainly been derived from distributed system mobility goals as well as some attributes that shape quality software. These cover transparency, security, robustness, consistency, synchronization, concurrency, visual adaptation, verification, validation, model support and many more.

##### 4.1 Reasoning Behind Selected Criteria

Complex mappings of agent concepts between analysis phase to the design phase needs to be kept at minimum. When this is achieved it enhances agent understandability, traceability as well as maintainability. In taking decisions about mobile agent in a specific system regarding the migration to/from a platform, the following should be taken into consideration; Robustness; reliability, security, performance and fault tolerance. Another issue is transparency. For agent to be extensible, open and fault tolerance, software designers must protect users from the issues of dealing with the actual platform of the service, concurrent access, and migration. To address some of these issues, concepts for mapping agent, concepts for describing agent's behaviour is needed. Concepts for dealing with communication, specifying and constraining agent migration are also needed. Concepts for proving pattern to help designers achieve transparency and other features must also be considered. For example, agent and mobile agents, internal and external structure and processes that some how relates to entities in complex problem. These issues are fundamental in open distributed system environment. Below are the criteria and a brief of each of them:

##### **Criterion 1:** Visual/ graphical modelling.

Visual or graphical modelling provides a natural way of expressing an idea. It also serves as a communication medium for expressing knowledge and idea throughout the development process. Formal or visual models all aims to describe real world system from a complex problem domain perspective.

##### **Criterion 2:** Consistency.

This criterion helps to preserves the models, agent concepts as the mobile agent transforms itself throughout the software processes.

##### **Criterion 3:** Verification.

This is very important in all phases of systems development to ensure that the correct product is being produced and the correct process is being followed.

##### **Criterion 4:** Validation.

It is equally important to carry out validation through all the development phases to ensure that the end product is usable in its intended environment and the operational needs are met.

##### **Criterion 5:** Well defined semantics and syntax.

Meanings of data for agents and how they are used must be consistent and unambiguous throughout all the phases of systems development. Set of rules to be applied during the exchanges of agent as it transforms itself through the phase should adhere to standard as defined.

##### **Criterion 6:** Well documented.

Complete and clearly defined systems documentation is required for maintenance. Documentation must be provided both at the systems level and user level. Systems documentation should outline the objectives of the entire systems, precise requirements as agreed between the developers and the stakeholders at the start of the project, how this requirement specification are implemented, how various agents and mobile agents interacts in the systems and their functions, and how all these requirements expressed in the chosen programming codes

##### **Criterion 7:** Mobility Support.

Modelling languages that models agent systems should also be able to model mobility.

##### **Criterion 8:** Static model support.

Modelling languages should be able to model static agents present in an environment or platform to show how these agents interacts and how they allocate resources when requested. Modelling languages should be able to model how these static agents reactive to their environment.

##### **Criterion 9:** Dynamic model support.

Modelling languages should be able to model the sequences of interactions between the agents and mobile agents from high level abstraction to low level abstraction. In other words, the transformation of agents and mobile agents should be supported as it transforms itself from requirement specification phase through to implementation phase of the development process. This includes the time the agent is created, executed and terminated, changes in interfaces and the mobility of other agent.

##### **Criterion 10:** Internal Structure Modelling.

Modelling Language should have the ability to model the internal state of the agent, what triggers it to move such as the goals and plan, why the agent decides to suspend, execute, terminate or move to another node or environment. In other words, the preconditions, invariant and post conditions should be modelled early in the requirement phase of the software development process. The sequences of execution should also be modelled. For an agent to reach a goal, the agent must have plan(s).

##### **Criterion 11:** External Structure Modelling.

Modelling languages should be able to model roles of agents and mobile systems, interface and interactions of agents within and outside their environment as well as complex interactions in the system.

##### **Criterion 12:** Case Study for Evaluation.

Case study plays a vital role in scientific research. Ability to target a specific problem and design task to evaluation cannot be over emphasised in the modelling languages. This gives researchers a basis for assessing the strengths and weaknesses and the variety of techniques of the modelling language.

##### **Criterion 13:** Extensible and Customizable.

Modelling languages should be flexible enough to accommodate new and additional words, phrases, stereotypes, grammar and rules. Modelling languages should have a mechanism to accommodate the mobility and dynamics of agents to suit different types of application. This is very important for agent adaptation and the changing needs of the environment and other communication agent.

#### Criterion 14: CASE Tools support.

Case tools support developers to analyse and design phases of software development process in the systems lifecycle. Integrated case tools will ensure agents communicate and interact together in providing solution to complex problems.

#### 4.2 Methodology for rating Agent Modelling under Criteria outlined in section 4.1

Rating of agent modelling language under the criteria was based on case studies and document sampling in published literature. We rated the criteria defined for existing agent modelling languages on a scale of 1 to 3 where 1- not at all supported, 2-partially supported 3- fully supported. Each criterion has been rated in comparison to the other modelling languages.

Table 1. Comparative Analysis.

CRITERIA MODELLING LANGUAGE	AML	ASL	AUML	SLABS	CAMLE
Visual/Graphical	3	1	2	1	2
Formal Structure	2	3	2	3	3
Consistency	3	2	2	3	3
Verification	1	1	1	3	3
Validation	1	1	1	3	3
Well defined semantics& syntax	3	2	2	3	3
Well documented	3	1	2	3	3
Extensible & customizable	3	1	2	1	1
Mobility support	2	1	2	1	2
Static model support	3	2	2	2	2
Dynamic model support	3	3	2	2	2
Internal structure	3	3	2	1	1
External structure	3	2	2	3	3
Case study for evaluation	1	1	1	1	3
CASE Tool Support	3	1	1	1	3

## 5. Discussion

Market forces drive the need for software houses to improve upon the efficiency in the design of cutting edge solutions and software products to support systems. Lapses in the requirement specification and analysis stage have created a huge gap and while this seems to be the most important phase crucial to the development of the product [8]. Balmelli [8] emphasises that this phase sees the transition of customer needs into product function and lack of support at this phase to realisation of product requirements. This in effect hinders the communication and understanding between stake holders and systems developers and modellers.

It is therefore imperative for modelling languages to be highly visual for communication between customers and developers. AML which draws its fundamental concepts from UML had addressed this one to some extent. CAMLE has models such as collaborative model which specifies the interaction of agents, behavioural models which specifies how agents' decisions are made, as well as partial support for graphics representation. Visual modelling has a powerful way of representing and communicating knowledge. It is one area that had been the back bone of success in Object Oriented (OO) Technology.

In OO Technology, CASE Tools have provided means of capturing business processes and also provided support for model construction as in companies such as British Airways [5]. Examples of CASE Tools being used in

the industry are Rational Rose and System Architect. To extract knowledge in CASE Tools one need to have knowledge of it and some level of understanding of graphical modelling. Modelling language therefore plays a vital role and as such is an important feature for automation. AML and CAMLE [3] provided a full support for CASE Tool automation.

Designing an application for safety critical systems, precision and correctness of the software is very important. Validation cannot also be overemphasised in all application area. To this end, SLABS and CAMLE has made ample provision to rigorously verify and validate all phases of systems development.

The development of software systems, sometimes lack effective formal representation of knowledge. This affects the ability to make decisions for large scale systems. SLABS and CAMLE provide facilities for representing high level concepts. It is apparent on table 1 and the appendices that modelling mobility is a challenge giving the issues other models have been able and not able to address.

It has therefore been relegated to the design phase. Modelling languages that attempted to model mobility from the table were AML and CAMLE and ASL.

ASL focuses on operational modelling of agent, fully support formal representation of agent, modelling of internal structure and support for dynamic models. AML comparatively provides support for modelling the external and internal structure of the agent, has a well defined semantics and syntax, as well as a full documentation for its processes.

AUML is an extension of the Unified Modelling language. Even though UML has advanced functionalities and advanced features in modelling objects, AUML is still at its infancy stage of language development and as such much needs to be done for it to reach maturity. There have been attempts to model mobility by languages such as CAMLE, AUML and AML but only at analysis and design phase of the development process. The timeline in figure 1 and appendices reiterate that modelling languages for modelling agent has not reached its maturity. There is still a lot to be done in validating the languages to optimise its usefulness in order to reap the benefits of modelling mobile agents.

## 6. Summary

Based on the complexity of software problems and the evolution of agent as an alternative for providing solution for complex problem, criteria for comparing the existing languages to ensure its relevance and usefulness in software development process were introduced based on their functions and features required for modelling agents with respect to mobility. Results from the comparative study presented shows that no single language possess all the functionality for modelling agent systems. In the same study the strengths and weaknesses of each of the languages has been established as shown in table 1. This means that a future hybridization is possible in order to model all phases of software development just as we now have for the unified modelling language.

## 7. Future Work



Future work will focus on an empirical study as a means of reassessing the rating of the criteria for each agent modelling language. We will also expand the scope of the criteria by looking specifically into goals of distributed mobility that we believe underpin requirements necessary for developing a methodology that captures mobility in software agents.

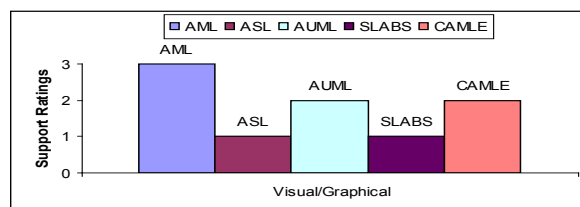
## References

- G. Weib, . Fischer, M. Nickles and M. Rovatsos, Operational modelling of Agent Autonomy: theoretical Aspects and Formal language. J.P. Muller and F. Zambonelli (Eds.): AOSE 2005, LNCS 3950, pp. 1-15. 2006.
- H. Mouratidis, J. Odell, and G. Manson. Extending the Unified Modeling Language to model Mobile Agents. In the Proceedings of the Agent Oriented Methodologies Workshop, OOPSLA 2002, Seattle - USA, November 2002
- H. Zhu and L. Shan., Caste-Centric Modelling of Multi-Agent Systems: The CAMLE Modelling Language and Automated Tools, in Beydeda, S. and Gruhn, V. (eds) Model-driven Software Development, Research and Practice in Software Engineering, Vol. II, Springer, 2005, pp57-89.
- H. Zhu, SLABS: A Formal Specification Language for Agent-Based Systems, International Journal of Software Engineering and Knowledge Engineering, Vol. 11. No. 5, pp529-558. Nov. 2001
- J. Arlow, J. Quinn, W. Emmerich. Literate Modelling-Capturing Business Knowledge with UML. In J. Bezivin and P.A. Muller, editors, Proc. UML'98, Mullhouse France, volume 1618 LNCS, pp. 165-172, Springer Verlag, 1999
- J.Odell, H.Parunak, B.Bauer, Extending UML for Agents. In the proceedings of Agent-Oriented Information systems Workshop at the 17th National Conference on Artificial Intelligence, 2000.
- I. Jacobson, G. Booch, J. Rumbaugh: The Unified Software Development Process, Addison Wesley, 1998.
- L. Balmelli, An Overview of Systems Modeling Language for product and Systems language for product and systems development. T.J. Watson Center and Tokyo Research Laboratory, IBM, 2006.
- L. Shan and H. Zhu, CAMLE: A Caste-Centric Agent-Oriented Modelling Language and Environment, Proc. of SELMAS'04 at ICSE'94, May 2004, Edinburgh, UK, IEE 2004, pp66-73.
- M. Kang, L. Wang and K. Taguchi. Modelling Mobile Agent applications in UML 2.0 Activity Diagrams, 519-522, ICEIS 2004, Proceedings of 6<sup>th</sup> International Conference on Enterprise Information Systems, Porto , Portugal, April 14-17, 2004.
- M. Wooldridge, Coherent social action. In proceedings of the Eleventh European Conference and Artificial Intelligence (ECAI-94), Amsterdam, The Netherlands, 1994.
- N. R. Jennings "Agent Oriented software Engineering" Proc. 12th International Conference on Industrial and Engineering Applications of AI, Cairo, Egypt, 4-10. Also appearing in Proc. 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-99), Valencia , Spain 1999.
- [1] R.Cervenka, I Trencansky, M.Calisti, D.Greenwood: AML Agent Modeling Language . Towards Industry Grade Agent -Based Modeling. In J.Odell, P.Giorgini, J.Muller, (eds): Agent Oriented Software Engineering V:

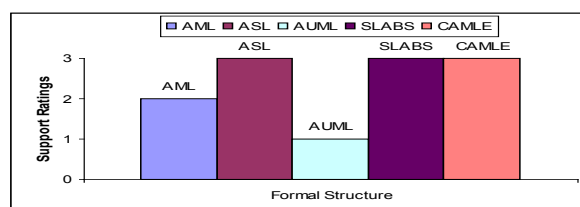
5th International Workshop, AOSE 2004, pp. 31 Springer-Verlag 2005.

## Appendix A

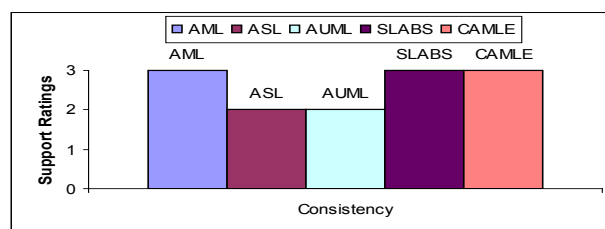
### A.1 Graphical representation for Visual/ graphical Support



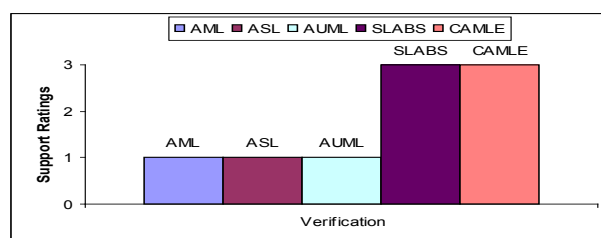
### A.2 Graphical representation for Formal Structure Support



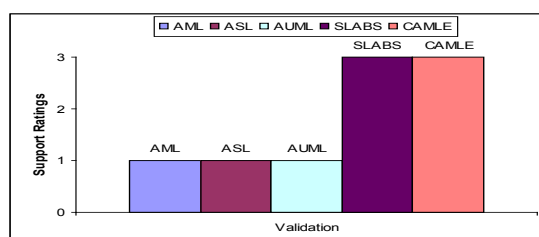
### A.3 Graphical representation for Consistency Support



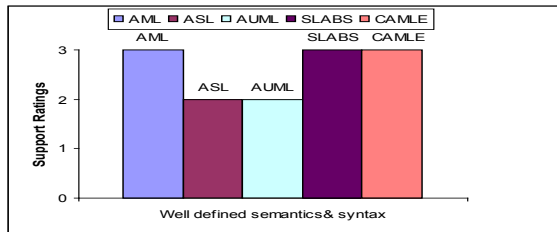
### A.4 Graphical representation for Verification Support



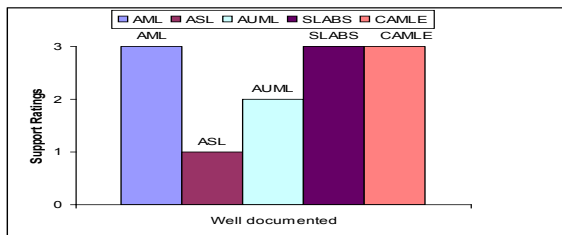
### A.5 Graphical representation for Validation support



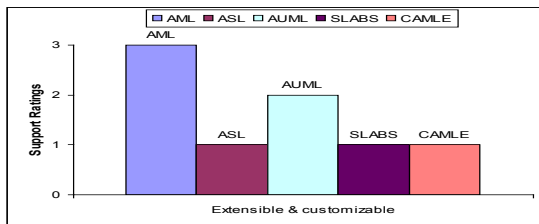
## A.6 Graphical representation for Well defined semantics & syntax



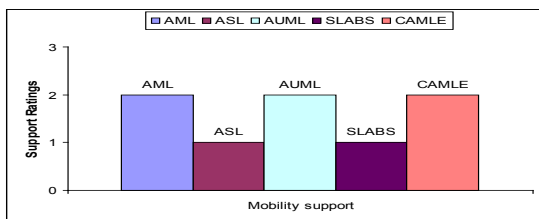
## A.7 Graphical representation for well documented



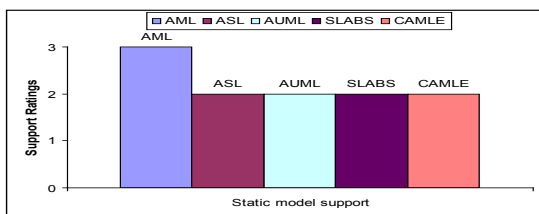
## A.8 Graphical representation for Extensibility and customizability support



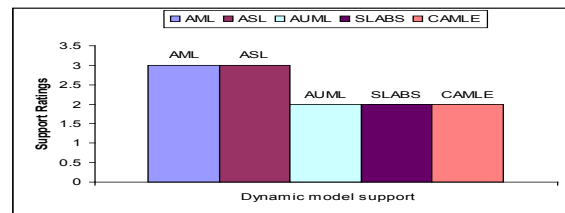
## A.9 Graphical representation for Mobility support



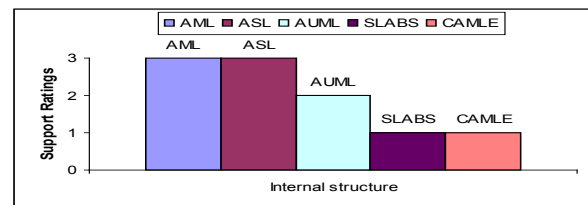
## A.10 Graphical representation for Static model support



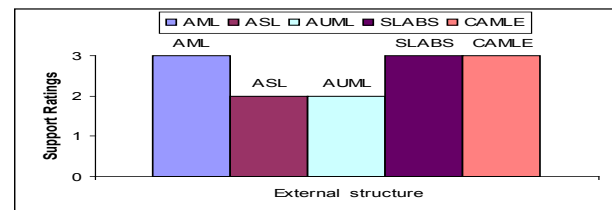
## A.11 Graphical representation for Dynamic model support



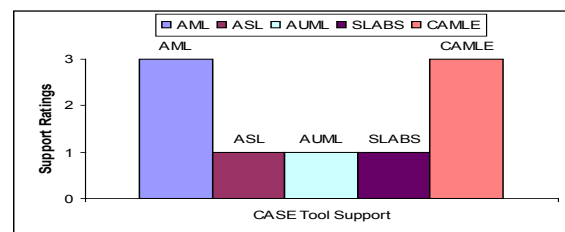
## A.12 Graphical representation for Internal Structure Support



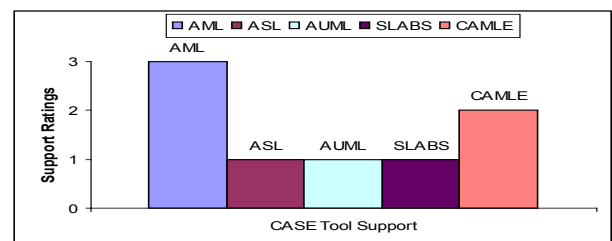
## A.13 Graphical representation for External Structure Support



## A.14 Graphical representation for Case Study for Evaluation



## A.14 Graphical representation for CASE Tool Support



# MOBILITY REQUIREMENTS ON GAME PLATFORMS: AN AGENT PERSPECTIVE

Divina Melomey, Godfried Williams and Chris Imafidon  
School of Computing and Technology  
Docklands Campus  
University of East London  
United Kingdom,  
E-mail: {divina,G.Williams,chris12}@uel.ac.uk

## KEYWORDS

Mobile Agents, Distributed Systems, Mobility,  
Game Platforms

## ABSTRACT

Game theory has been used as a tool in the past few years to address several social issues and also to provide explanation to resource based problems in academia. Games are used to simulate real life events where an individual pursues an interest either selfishly to the detriment of others or collaboratively. This sometimes can lead to conflicts or sometimes fierce competitions. Competition in general, leads to a group or an individual adopting a strategy to win. Elements of competition for mobility resources on game platforms lead to constraints that have to be managed and optimized effectively. *This paper evaluates mobility requirements for mobile agents on game platforms and proposes a first level generic model for capturing such requirements.*

## 1.0 INTRODUCTION

Autonomous agents and multi agents have become very important in research and development. These paradigms draw concepts from distributed computing, object oriented systems, software engineering, artificial intelligence, economics, game theory, sociology and organizational science. This concept offers solutions to complex software systems through analyzing, designing and implementing them (Jennings et al. 1998).

Jennings

(2000) identified Agent Oriented Systems Engineering (AOSE) as having the potential of improving considerably the practices of software engineering. This is because the concepts of AOSE are complementary to providing software solutions to complex systems. They attempted to give reasons why in certain problems, the best way of solving them is by adopting a multi agent approach to systems development. There is no general consensus definition for what an agent is but there are general characteristics by which an agent could be identified. Agents are characterized by autonomy, social/interactive, proactive/goal oriented, reactive, persistent and a desirable property such as mobility, adaptation and rationality (Brustoloni 1991) (Smith et al.1994) (Wooldridge and Jennings 1995)( Franklin and

Graesser, 1996), (Melomey and Mouratidis 2006) ,(Williams G.B. 2007a). (Melomey et al. 2007a)(Melomey et al. 2007b).

## 1.1 MOBILE AGENTS

A mobile agent is an autonomous software program that can migrate from one platform to another on a heterogeneous network performing task on behalf of the user (Milojicic, 1999). It is a computational process that implements the autonomous, communicating functionality of an application and is able to migrate from one computer to the other over a network. The platform is made up of the computational environment and the agent is also made up of the code and State information that is needed to perform some form of computation (Cubaleska and Schneider, 2002). In other words, the platform provides the physical environment for deployment of the agent; an agent can be said to have set of attribute called state which describes its characteristics. Agents communicate via an Agent Communication Language (ACL).

Jansen (2002) defined mobile agent as “traveling agents”, these programs will shuttle their being, code and state, among resources.”

In this research we define mobile agent as autonomous agent that exhibit mobility characteristics such as persistency, robustness, security assessment for its codes and environment, mobility transparency and fault tolerance.

We also define a mobile agent as program that exhibits persistency, fault tolerance, synchronization, remote addressing and referencing, calling , invocation , execution, remote code execution and migration capabilities.

A mobile agent on a game platform should satisfy certain requirements to enable it migrate from one platform to another, as well as exercising mobility via remote access during the period in which a game is in session or being played. *This paper evaluates generic mobility requirements for distributed platforms with specific emphasis on game platforms and game applications that exhibit social adaptation, intelligence, collaboration, autonomy as well as key features peculiar to mobile agents.*

## 2.0 GENERIC REQUIREMENTS OF DISTRIBUTED PLATFORMS

Distributed systems platforms have requirements that are generic to all platforms for all types of distributed applications. All applications must satisfy these requirements. These requirements are resource sharing, openness, concurrency transparency, scalability, transparency and fault tolerance (Galli 1999).

### **Resource Sharing**

The platform controls all resources including allocation and access control and concurrency. A resource manager is allocated the responsibility for sharing resources anywhere on the system and also interactive activities on the platform.

### **Platform Openness**

Platform openness involves enabling the integration with existing components by adding new ones, publishing component interfaces, resolutions of interfaces issues relating to heterogeneous processors in the distributed environment.

### **Concurrency**

Concurrency allows accesses and updates of shared resources, without which the integrity of the systems might be compromised. Executions of components are done in concurrent processes.

### **Scalability**

Distributed system platform allows more users to be included and adapts quickly to its environment. It is achieved by adding faster processors to accommodate the new additions hence scalability. Component must therefore be designed to be scalable.

### **Fault Tolerance**

All networks, software and hardware are susceptible to breakdown hence any distributed system platform must be design to be able to recover after a breakdown. Fault tolerance on a platform maintains a certain level of reliability for such systems and achieved through recovery and redundancy.

### **Transparency**

Transparency on distributed systems platform makes information available for access whether it is remote, location base, migration, scalability, concurrency, performance or failure without any interference.

### **Mobility**

Another key requirement of distributed System Platform is mobility. Mobility in distributed systems is demonstrated through both physical and logical migration (Roman et al 2000). In physical migration, a process or program travels across the physical network from node to node or server to server via designated network routes' using the IP addresses system. In logical migration, there is remote execution of processes and programs through a remote procedure calling or remote method invocation system. This is achieved at the back of Client Server Stubs implemented on the system.

The next session will discuss specific requirements of mobility in a distributed systems platform.

## **3.0 MOBILITY REQUIREMENTS ON DISTRIBUTED SYSTEMS**

Mobility requirement are key design elements that needs to be satisfied in distributed systems applications. Distributed platforms should be configured to have the ability to monitor and control resources per client request as well as the activities of clients on the platform. Another requirement is the ability to identify the location of the client at all times and the hardware on which the resource being accessed is located. There is the need for entities/components and distributed platform including local and remote platform to trust each other in order to share and access resources. This means that security of both platforms and agent's platforms both static and mobile must not be compromise.

For effective use of resources in a distributed platform, resource and location must be available for all distributed applications (Spence et al. 2005). This is essential for users to experience a low latency rate and minimum network or communication failure. In order to ensure reliability of distributed network as well as latency, the location of application components must be taken in to consideration during platform design. Resource must be available and close to cluster of users to ensure there are minimum delay propagation delays, maximum throughput and also minimum network failures. Foster et al (2002) also identified some common requirements relating to delivery of service common to distributed system mobile environments. These were security semantics, resource management, distributed work flows, fault tolerance and problems determination services and other metrics that are unique to an individual application yet important as a requirement in a distributed environment.

*Other issues in the implementation of a distributed system are addressing, encoding and synchronization. If components on a distributed environment have to locate each other, both client and server must have address. This section of the paper summarizes **Mobility Requirements on Distributed Systems, covering addressing, encoding, synchronization, persistent, invocation, calling, naming and message passing.***

### **Addressing**

Servers must broadcast their address by making it reachable to clients for access to resources and services. This is usually achieved by looking it up in a look up table in a naming service or server or a server's registry. Addressing is made up of the name of the host and port number of the host (Henning, 1998). For example a server makes itself available by binding itself to a port in order to be contacted by a client. Servers and clients may not necessarily be located physically together hence the type of communications between them could either be in the form of message passing or data streaming. Communication will only be successful when host names together with port numbers to be used are agreed by participating parties on the distributed platform.

## Encoding

This is a formatting technique for streams of data being transferred. This helps developers to structure complex data (Neema et al 2003). Encoding may be used to transform data streams based on specific application. This can be achieved through pre processed data on hardware component that is not running a core application. This hardware component will usually be dedicated for executing streams of transmitted data (Gavrilovska et al 2005)

## Synchronization

“Synchronization is mainly to ensure that times, associated and recorded with respect to the occurrence of network events are consistent and valid” (Williams G. B 2007b). Synchronization takes place when participating parties that is clients and servers on the platform agree to some level of protocol agreement regarding sending and receiving of data. These transfer data protocols could pertain to how messages are relayed or requested. Components are reached via a known location and data is exchanged usually using a predefined set of communication protocol. In other words there are governing rules that enable synchronisation. These rules are usually presented in the literature as algorithmic.

## Persistence

Persistence entities permit communications directly between the server and the client. Persistent entities allows for direct communication between client and server. For example if the server's transport end point is disconnected as a result of server shutdown, all the client's references into that server become invalid, and no further communication is possible. Persistent entities are able to recover from such a shut down or systems breakdown because they have a state (Spencer 1996).

## Method Invocation

Methods can be invoked and executed remotely when interface and address information are known. Method invocation depends largely on method. There are two types of invocation and they are local and remote invocation. With the local invocation a local entity is passed by copying using a standard object serialization while in the remote invocation it is a remote object that is passed by reference to its proxy. Method invocation can be implemented remotely in two ways: Remote Method Invocation and Common Object Oriented broker Architecture (CORBA). RMI is java only distributed object model and easy to use. It is also able to integrate with CORBA. RMI minimizes the differences working with local and remote entities. Secondly, it minimizes the complexity of asks while supporting distributed garbage collection. CORBA on the other hand, is language independent thus it can be written in any language. CORBA is an OMG standard and more matured hat RMI hence has capabilities well defined Colouris el al (2001), WilliamsG B (2000).

## Interface Definition Language

It is a language that is used to describe the interface of a local or remote interface. Interface definition language are basically used to declare constructs used to export methods and further made available to clients. Invoked methods usually have specific typed parameters and return values. Parameters could be strings or numeric types. These declarations are usually independent of programming language employed after which it is compiled by an interface definition language compiler to produce declarations that are required by a specific language. OMG defines it as a specification language uses a common set of data types used for defining complex data types.

## Naming Services

Entities have names by which they are known. Entities are mapped to their names and location. Entities in distributed platform shares and exchange data among themselves hence there is a need for a service mechanism to be responsible for creating, naming and managing these entities independently. Name servers manage all name information and name hierarchies. Naming service is autonomous and indispensable feature for persistent and transparency of an entity. Naming services allows entities to identify each other and the location where an entity originates from. According to Yeo et al (1993) naming services is divided into three parts. They are whit pages for mapping symbolic names to network addresses, yellow pages which provide directory services which provide support for searches that are based on the description of software entities, and lastly broadcast based discovery useful for locating entities on a local area network (Mockapetris and Dunlap 1988) (Postel and Anderson 1994).

## 4.0 NATURE OF GAME PLATFORMS

The underpinning architecture of game platforms are distributed in nature, according to the research conducted by the authors. Overmars M(2004), in the paper game maker uses object oriented event driven approach for game maker application. The initial observation made is that mobility on distributed platforms could be achieved via inheritance, polymorphism and abstractions. These characteristics of objects enable mobility. This proposition is subject to further investigation and further analysis. Hiromichhi et al (2004), emphasize on components base development approach. The system described in their paper “3D visual component based voice input and output interfaces for interactive development” highlights the use of intelligent boxes that contain 3D objects. According to this paper the essential aspect of the box is a component known as the model-display object (MD) structure. This component is made up of two objects, the controller and viewer (MVC) structure. The states value of a model is held in a box. Variable spaces called slot store these states. It is important to note that components based platforms are distributed based.

Fundamentally this platform employs the TCP/IP protocol stack for Client-Server interaction or socket communication. There is also an indication of clear messaging between boxes that form the base architecture. These stores the state values of a box. The component nature of the platform allows plug-in application such as "Microsoft Speech API". Similar game applications have also demonstrated the need for effective distributed systems in supporting game applications.

Bancroft M & Al-Dabass D,(2004) also employ visual C++ a Microsoft OO language in the development of their game platform. What is not however clear is whether, the language was chosen because of the author's familiarity or the object nature of the language. One thing which strikes us is the fact that Visual C++ enabled the game to be deployed effectively. It is important to note that the nature of the application needs to satisfy mobility requirements.

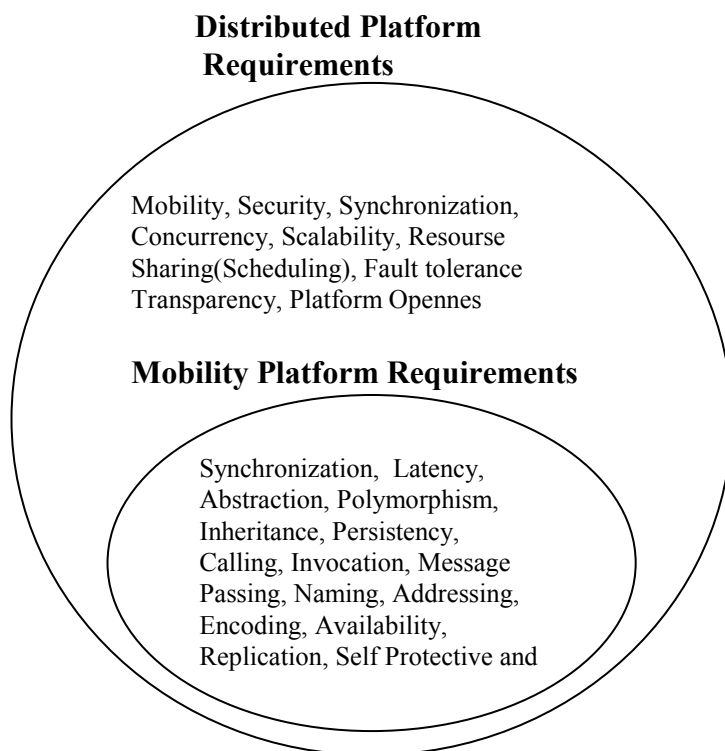
Zeng X, Mehdi Q.H, and Gough (2004) describe the implementation of a game platform using VRML and JAVA for visualization tasks. The paper focuses discussions on a visualizer graphic engine (architecture). Their work indicates that event sending aspects of VRML could be deemed as a strong characteristic of mobility within the infrastructure. VRML allows interactivity in real time. Their paper explores the feasibility of binding VRML and JAVA to provide real time communication. The notion here is that object based technologies play pivotal role in building interactive game platforms. VRML provides virtuality while JAVA based technologies facilitate communication on distributed platforms. It is essential also to note from this work that sociological issues highlighted stamp out the need to appreciate and understand dynamicity of mobile interactions. The need for mobile systems to occupy time and space, highlights the need for mobility. Other researchers such as Simatic M et al (2004) in their work "technical and usage issues for mobile multiplayer games" highlights issues relating to communication middleware prototype compliant to Open Mobile Alliance specifications. They also examined the work of "Group des 'Ecoles des Telecommunication" known as MEGA (MultiplayEr Games Architecture). According to them, the common issues with mobile multi-player games are abstraction, latency, consistency and databases (DBMS). These could also be considered as essential mobility requirements.

Thorn D, Slater D(2004) also discuss things to consider when developing distributed adventure games by examining platforms and technologies available for the development of MMORPG(Massively Multi-player Online Role Playing Games). There is a discussion on potential technologies that are likely to help accelerate development in that area. Common components of platforms include SMS Server (SMS Technology). MMS (Multi-Media Messaging). LBS (Location Based Services), usage of GSM cells help to locate players in different communities. There is also the use of GPS satellite with custom made receivers' Short range positioning beacons (SRPB) uses Wi-Fi connections and blue tooth technology Williams(~2007).

Solinger D, Ehlert P Rothkrantz (2005) describe autonomous agent that controls airplane dog fights. Dogfights agent provides independent reasoning during artificial piloting. This is based on the intelligent Cockpit environment (ICE). The architecture for this application comprise of MCFS (Microsoft Combat Flight Simulation) interacting via the TCP/IP protocol. The system is implemented using visual C++ each object in the agent architecture is implemented using C++ class. The work of Bouillot N, (2005) fast event ordering and perspective consistency in time sensitive distributed multi-player games emphasize usefulness of consistency model, as a means of ensuring synchronization. This also brings to light that consistency, synchronization contribute to enabled and effective mobility on distributed platforms.

## 5.0 ENABLING AGENT MOBILITY ON GAME PLATFORMS

The distributed nature of game platforms as exposed in this investigation underpins and highlights the fact that mobile agents deployed on game platforms need to satisfy certain key characteristics in order for agents to exhibit mobility. This work provides new insights and directions necessary in capturing essential requirements when designing mobile agents for distributed system applications such as computer and internet games. According to our findings mobility platforms requirements can be classified into four main groups, these are; 1. **Timing requirements** - Latency (response times) and Synchronization 2. **Behavioral requirements** - Polymorphism, Inheritance, Persistency, Calling, Invocation, location, message passing; 3. **Addressing requirements** - Location, Naming and Encoding, 4. **Security requirements** - Availability, Self Protective, Fault tolerance and Certified Figure 1 summarizes generic requirements for distributed and mobility platforms. According to our study mobility



**Figure 1 -  
Generic Distributed Mobility Platform**

## 6.0 CONCLUSIONS

Our work for the first time has classified mobility requirements for distributed based applications into the following four main categories.

1. **Timing requirements** - Latency (response times) and Synchronization 2. **Behaviorial requirements** - Polymorphism, Inheritance, Persistency, Calling, Invocation, location, message passing; 3. **Addressing requirements** - Location, Naming and Encoding, 4. **Security requirements** - Availability, Self Protective, Fault tolerance, Replication and Certified Figure 1 summarizes generic requirements for distributed and mobility platforms. According to our study of mobility, the authors believe that these are critical success factors that a mobile agent has to satisfied in order to exhibit effective and efficient mobility on distributed platforms.

## 7.0 REFERENCES

Brustoloni, J.C. 1991. "Autonomous Agents: Characterization and Requirements." Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University

Bancroft M. and D. Al-Dabass 2004. "A Combat Simulation Aid for Dungeon and Dragons". In *Proceedings of 5th Game-On International Conference*, pp 60-65. Reading, UK.

Cubaleska, B. and M. Schneider 2002 "Detecting DoS Attacks in Mobile Agent Systems and using Trust Policies for their Prevention", Policy Workshop, International Workshop on policies for Distributed Systems and Networks POLICY 2002: 198-201

Foster I., C. Kesselman, J.Nick, S.Tuecke 2002. "Grid Services for Distributed System Integration Computer." IEEE, 35(6).

Fukutake H., Y. Okada and K. Nijima 2004. "3D Visual Component based voice on input/output interfaces for interactive graphic applications." In *Proceedings of 5th Game-On International Conference*, pp 20-24.

Franklin S. and A. Graesser 1996. "It an Agent, or just a Program?: A Taxonomy for Autonomous Agents." *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, Institute for Intelligent systems University of Memphis.

Galli D.L. 1999. "*Distributed Operating Systems: Concepts and Practice*". Prentice Hall, 1<sup>st</sup> Edition

Gavrilovska A., S. Kumar., S.Sundaragopalan., and K. Schwan 2005. "Platform Overlays: Enabling In-Network Stream Processing in Large-scale Distributed Applications", In *Proceedings of 15<sup>th</sup> International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV* (Skamania, Washington, Jun 13-15). *Communications of the ACM* : 1-58113-987

Henning M. 1998 " Binding, Migration, and Scalability in CORBA." *Communications of the ACM*, Vol. 41, No 10 (Oct.)

Jansen W. 2002. " Intrusion Detection with Mobile Agents". *Computer Communications*, Special Issue on Intrusion Detection Systems, vol. 25, num. 4, September.

Jennings N. R. 2000. "On Agent-Based Software Engineering" *Artificial Intelligence Journal* 117 (2) 277-296.

Jennings N. R., K. Sycara and M. Wooldridge 1998. "A Roadmap of Agent Research and Development" *International Journal of Autonomous Agents and Multi-Agent Systems* 1 (1) 7-38.

Melomey. D., and Mouratidis, H. (2006). "Evaluating the Security of Mobile Agent Platforms". In *Proceedings 1st Annual Conference on Advances in Computing and Technology (ACT'2006)* (London, United Kingdom, 24th January), pp. 48 -54.

- Melomey, D., H. Mouratidis, H. and C. Imafidon (2007). "An Evaluating the Security of Current Approaches for Modelling Mobility of Agent". In *Proceedings 2<sup>nd</sup> Annual Conference on Advances in Computing and Technology (ACT'2007)*, London, United Kingdom, 24th January, pp. 71-78.
- Melomey D., C. Imafidon and G. Williams(2007). "A Comparative Study of Modelling Languages for Agent Systems". Vol 1, No 2, (Jul), pp 207-212
- Milojicic D., D. Kotz, D. Lange , C. Petrie, C. Rygaard 1999. "Mobile agent applications" IEEE Concurrency (July to September) pp 80-90
- Mockapetris, P., K. Dunlap 1988 "Development of the Domain Name System". *Communications of ACM SIGCOM*, Stanford, CA, pp. 123-133.
- Neema S., J. Sztipanovits., G. Karsai. And K. Butts 2003. "Constraint-Based Design-Space Exploration and Model Synthesis." *Proceedings of Third International Conference on Embedded Software (EMSOFT)* , LNCS 2855, pp. 290-305.
- Overmars M. 2004. "Game Design in Education." In *Proceedings of 5th Game-On International Conference*, pp 9-18. Reading,UK.
- Postel, J., C. Anderson, 1994. "White Pages Meeting Report", RFC 1588, (Feb.)
- Roman, G.-C., G.P Picco, and A.L.Murphy, 2000. "Software Engineering for Mobility: A Roadmap,"In *Proceedings of 22nd International Conference on Software Engineering Future of Software Engineering*, A. Finkelstein (ed.), (invited paper) pp. 241-258.
- Simatic M., S. Craipeau, A. Beugnard, S. Chabidon, M-C Legout, E. Gressier. " Technical and Usage issues for Multiplayer games". In *Proceedings of 5th Game-On International Conference*, pp 134-138. Reading,UK.
- Smith, D. C., A. Cypher and J. Spohrer 1994, "KidSim: Programming Agents Without a Programming Language," *Communications of the ACM*, 37, 7, 55-67
- Spence D., J. Crowcroft, S. Hand. and T.Harris 2005. "Location Based Placement of Whole Distributed Systems." In *Proceedings of the ACM conference on Emerging networking experiments and technologies*. PP 124—134.
- Thorn D., I. Palmer and E. Williams 2004. MMORG on Mobile Devices? Considerations When designing distributed Adventure games". In *Proceedings of 5th Game-On International Conference*, pp 150-154. Reading,UK.
- Williams G.B 2000 "Technical Notes in RMI" Achival & Unpublished Work
- Williams G.B 2007a. "Artificial Intelligence" Existing & Emering Techniques" Google 1<sup>st</sup> Edition
- Williams G.B 2007b. " Online Business Security Systems". Springer , 1<sup>st</sup> Edition.
- Wright T. 2004. "Naming Services in Multi-Agent Systems: A Design for Agent-Based White Pages". In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems- AAMAS* pp 1478-1479
- Yeo, A., Ananda, A., Koh, E. 1993 "A Taxonomy of Issues in Name Systems Design and Implementation", *Communications of ACM SIGOPS Operating Systems Review*, New York, NY, July , pp. 4-18
- Zeng X., Q.H. Mehdi and N.E. Gough 2004. "Implementation of VRML and Java for Story Visualisation Tasks". In *Proceedings of 5th Game-On International Conference*, pp 112-126. Reading,UK.



# A FITNESS FUNCTION FOR CAPTURING MOBILE AGENT MOBILITY ON GAME PLATFORMS

Divina Melomey, Godfried Williams, Chris Imafidon, Roy Perryman  
School of Computing and Technology  
Docklands Campus  
University of East London  
United Kingdom,  
E-mail: {divina,G.Williams.chris12}@uel.ac.uk

## KEYWORDS

Fitness Function, Genetic algorithm, Mobile Agents, Distributed Systems, Mobility, Mobile Game Platforms

adaptation and rationality (Brustoloni 1991) (Smith et al.1994) (Wooldridge and Jennings 1995)( Franklin and Graesser, 1996), Williams G.B.(2007a). Melomey D(2007), Melomey D (2006).

## MOBILE AGENTS

A mobile agent is an autonomous software program that can migrate from one platform to another on a heterogeneous network performing task on behalf of the user (Milojicic, 1999). It is a computational process that implements the autonomous, communicating functionality of an application and is able to migrate from one computer to the other over a network. The platform is made up of the computational environment and the agent is also made up of the code and State information that is needed to perform some form of computation (Cubaleska and Schneider, 2002). In other words, the platform provides the physical environment for deployment of the agent; an agent can be said to have set of attribute called state which describes its characteristics. Agents communicate via an Agent Communication Language (ACL).

Janson (2000) defined mobile agent as “traveling agents”, these programs will shuttle their being, code and state, among resources.”

A mobile agent as autonomous agent that exhibit mobility characteristics such as persistency, robustness, security assessment for its codes and environment, mobility transparency and fault tolerance Melomey D et al (2007).

“A mobile agent on a game platform should satisfy certain requirements to enable it migrate from one platform to another, as well as exercising mobility via remote access during the period in which a game is in session or being played” Melomey et al (2007). *This paper evaluates generic mobility requirements for distributed platforms with specific emphasis on game platforms and game applications that exhibit social adaptation, intelligence, collaboration, autonomy as well as key features peculiar to mobile agents.*

## ABSTRACT

Genetic algorithm has been used to model problem scenarios where there is the need to optimize resources. Mobile games require effective strategies to ensure remote access to data and processes in a transparent manner. The criteria necessary for capturing and modeling mobility in such systems can be ambiguous and cumbersome, if an effective selection criteria is not adopted. The deployment of mobile agents in mobile games require the use of a selection criteria for capturing and modeling mobility in a manner that make effective use of available features of system resources optimally. This work explores and exploits a fitness function using genetic algorithm as a criteria for selecting requirements and characteristics key to modeling mobility for a mobile agent deployed on a mobile game platform.

## INTRODUCTION

Mobility requirements on mobile game platforms causes the need to have an intelligent program function that utilizes system resources effectively. The application of mobile agents serves as a driver for meeting this functional requirement. It has also become an essentially criteria for achieving mobility as a functional goal. The paradigms underlying mobile agents is drawn from distributed computing, object oriented systems, software engineering, artificial intelligence, economics, game theory, sociology and organizational science concepts Melomey et al (2007).

Jennings (2000) described Agent Oriented software Engineering (AOSE) as essential to improving software engineering, given that AOSE complements software solutions for complex systems. Multi agent approach to systems development is also becoming important in systems development. There is no general consensus definition for what an agent is but there are general characteristics by which an agent could be identified. Agents are characterized by autonomy, social/interactive, proactive/goal oriented, reactive, persistent and a desirable property such as mobility,

## COMMON DISTRIBUTED SYSTEM REQUIREMENTS

Distributed systems platforms have requirements that are generic to all platforms for all types of distributed applications. All applications must satisfy these requirements. These requirements are resource sharing, openness, concurrency transparency, scalability, transparency and fault tolerance (Galli 1999).

Resource Sharing, Platform Openness, Concurrency, Scalability, Fault Tolerance, Transparency, Mobility (Melomey et al (2007)).

## MOBILITY REQUIREMENTS ON DISTRIBUTED SYSTEMS

Mobility requirements are key design elements that need to be satisfied in distributed systems applications. Distributed platforms should be configured to have the ability to monitor and control resources per client request as well as the activities of clients on the platform. Another requirement is the ability to identify the location of the client at all times and the hardware on which the resource being accessed is located. There is the need for entities/components and distributed platform including local and remote platform to trust each other in order to share and access resources. This means that security of both platforms and agent's platforms both static and mobile must not be compromised.

For effective use of resources in a distributed platform, resource and location must be available for all distributed applications (Spence et al. 2005). This is essential for users to experience a low latency rate and minimum network or communication failure. In order to ensure reliability of distributed network as well as latency, the location of application components must be taken into consideration during platform design. Resource must be available and close to cluster of users to ensure there are minimum delay propagation delays, maximum throughput and also minimum network failures. Foster et al (2002) also identified some common requirements relating to delivery of service common to distributed system mobile environments. These were security semantics, resource management, distributed work flows, fault tolerance and problems determination services and other metrics that are unique to an individual application yet important as a requirement in a distributed environment.

*Other issues in the implementation of a distributed system are addressing, encoding and synchronization. If components on a distributed environment have to locate each other, both client and server must have address. This section of the paper summarizes **Mobility Requirements on Distributed Systems**, covering addressing, encoding, synchronization, persistent, invocation, calling, naming and message passing.*

## Addressing

Servers must broadcast their address by making it reachable to clients for access to resources and services. This is usually achieved by looking it up in a look up table in a naming service or server or a server's registry. Addressing is made up of the name of the host and port number of the host (Henning, 1998). For example a server makes itself available by binding itself to a port in order to be contacted by a client. Servers and clients may not necessarily be located physically together hence the type of communications between them could either be in the form of message passing or data streaming. Communication will only be successful when host names together with port numbers to be used are agreed by participating parties on the distributed platform.

## Encoding

This is a formatting technique for streams of data being transferred. This helps developers to structure complex data (Neema et al 2003). Encoding may be used to transform data streams based on specific application. This can be achieved through pre processed data on hardware component that is not running a core application. This hardware component will usually be dedicated for executing streams of transmitted data (Gavrilovska et al 2005).

## Synchronization

"Synchronization is mainly to ensure that times, associated and recorded with respect to the occurrence of network events are consistent and valid" Williams G B (2007)b.

Synchronization takes place when participating parties that is clients and servers on the platform agree to some level of protocol agreement regarding sending and receiving of data. These transfer data protocols could pertain to how messages are relayed or requested. Components are reached via a known location and data is exchanged usually using a predefined set of communication protocol. In other words there are governing rules that enable synchronisation. These rules are usually presented in the literature as algorithmic.

## Persistence

Persistence entities permit communications directly between the server and the client. Persistent entities allow for direct communication between client and server. For example if the server's transport end point is disconnected as a result of server shutdown, all the client's references into that server become invalid, and no further communication is possible. Persistent entities are able to recover from such a shut down or systems breakdown because they have a state (Spencer 1996).

## Method Invocation

Methods can be invoked and executed remotely when interface and address information are known. Method invocation depends largely on method. There are two types of invocation and they are local and remote invocation. With the local invocation a local entity is passed by copying using a standard object serialization while in the remote invocation it is a remote object that is passed by reference to its proxy. Method invocation can be implemented remotely in two ways: Remote Method Invocation and Common Object Oriented broker Architecture (CORBA). RMI is java only distributed object model and easy to use. It is also able to integrate with CORBA. RMI minimizes the differences working with local and remote entities. Secondly, it minimizes the complexity of tasks while supporting distributed garbage collection. CORBA on the other hand, is language independent thus it can be written in any language. CORBA is an OMG standard and more matured than RMI hence has capabilities well defined Colouris et al (2001), Williams G B (2000).

## Interface Definition Language

It is a language that is used to describe the interface of a local or remote interface. Interface definition language are basically used to declare constructs used to export methods and further made available to clients. Invoked methods usually have specific typed parameters and return values. Parameters could be strings or numeric types. These declarations are usually independent of programming language employed after which it is compiled by an interface definition language compiler to produce declarations that are required by a specific language. OMG defines it as a specification language uses a common set of data types used for defining complex data types.

## Naming Services

Entities have names by which they are known. Entities are mapped to their names and location. Entities in distributed platform shares and exchange data among themselves hence there is a need for a service mechanism to be responsible for creating, naming and managing these entities independently. Name servers manage all name information and name hierarchies. Naming service is autonomous and indispensable feature for persistent and transparency of an entity. Naming services allows entities to identify each other and the location where an entity originates from. According to Yeo et al (1993) naming services is divided into three parts. They are white pages for mapping symbolic names to network addresses, yellow pages which provide directory services which provide support for searches that are based on the description of software entities, and lastly broadcast based discovery useful for locating entities on a local area network (Mockapetris and Dunlap 1988) (Postel and Anderson 1994).

## GAME PLATFORMS

The core architecture of game platforms are distributed in nature, according to the research conducted by Overmars M(2004), object oriented was key to the event driven approach for game maker application. Our analysis revealed that mobility on distributed platforms could be achieved through inheritance, polymorphism and abstractions. These characteristics of objects enable mobility on game platforms. Hiromichhi et al (2004), emphasize on components base development approach. The system described in their paper “3D visual component based voice input and output interfaces for interactive development” highlights the use of intelligent boxes that contain 3D objects. According to this paper the essential aspect of the box is a component known as the model-display object (MD) structure. This component is made up of two objects, the controller and viewer (MVC) structure. The states value of a model is held in a box. Variable spaces called slot store these states. It is important to note that components based platforms are distributed based.

Fundamentally this platform employs the TCP/IP protocol stack for Client-Server interaction or socket communication. There is also an indication of clear messaging between boxes that form the base architecture. These stores the state values of a box. The component nature of the platform allows plug-in application such as “Microsoft Speech API”. Similar game applications have also demonstrated the need for effective distributed systems in supporting game applications.

Bancroft M & Al-Dabass D,(2004) also employ visual C++ a Microsoft OO language in the development of their game platform. What is not however clear is whether, the language was chosen because of the author’s familiarity or the object nature of the language. One thing which strikes us is the fact that Visual C++ enabled the game to be deployed effectively. It is important to note that the nature of the application needs to satisfy mobility requirements.

Zeng X, Mehdi Q.H, and Gough (2004) describe the implementation of a game platform using VRML and JAVA for visualization tasks. The paper focuses discussions on a visualizer graphic engine (architecture). Their work indicates that event sending aspects of VRML could be deemed as a strong characteristic of mobility within the infrastructure. VRML allows interactivity in real time. Their paper explores the feasibility of binding VRML and JAVA to provide real time communication. The notion here is that object based technologies play pivotal role in building interactive game platforms. VRML provides virtuality whiles JAVA based technologies facilitate communication on distributed platforms. It is essential also to note from this work that sociological issues highlighted stamp out the need to appreciate and understand dynamicity of mobile interactions. The need for mobile systems to occupy time and space, highlights the need for mobility. Other researchers such as Simatic M et al (2004) in their work “technical and usage issues for mobile multiplayer games” highlights issues relating to communication middleware prototype compliant to Open Mobile Alliance specifications.

They also examined the work of “Group des ‘Ecoles des Telecommunication” known as MEGA (MultiplayEr Games Architecture). According to them, the common issues with

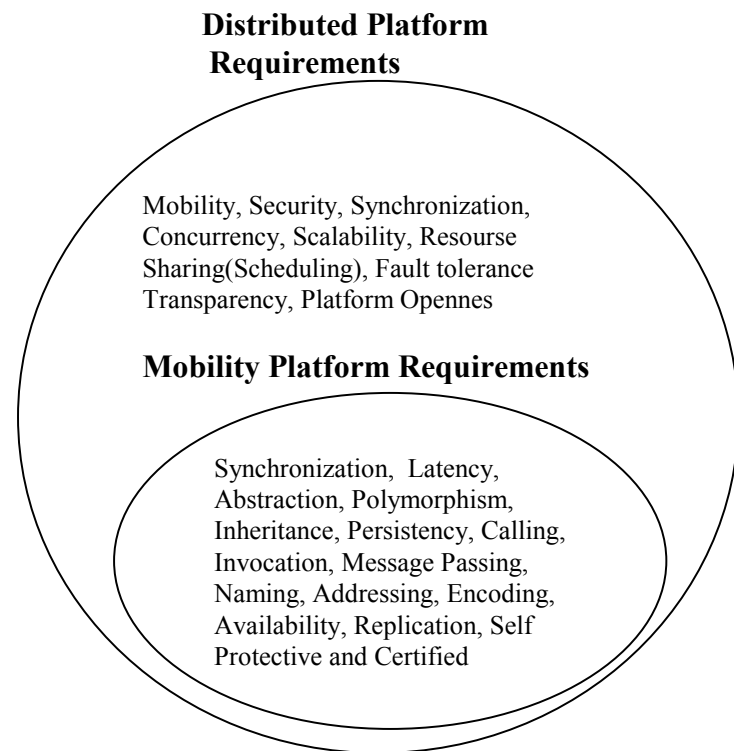
mobile multi-player games are abstraction, latency, consistency and databases (DBMS). These could also be considered as essential mobility requirements.

Thorn D, Slater D (2004) also discuss things to consider when developing distributed adventure games by examining platforms and technologies available for the development of MMORPG (Massively Multi-player Online Role Playing Games). There is a discussion on potential technologies that are likely to help accelerate development in that area. Common components of platforms include SMS Server (SMS Technology), MMS (Multi-Media Messaging), LBS (Location Based Services), usage of GSM cells help to locate players in different communities. There is also the use of GPS satellite with custom made receivers' Short range positioning beacons (SRPB) uses Wi-Fi connections and blue tooth technology Williams (~2007).

Solinger D, Ehlert P Rothkrantz (2005) describe autonomous agent that controls airplane dog fights. Dogfights agent provides independent reasoning during artificial piloting. This is based on the intelligent Cockpit environment (ICE). The architecture for this application comprise of MCFS (Microsoft Combat Flight Simulation) interacting via the TCP/IP protocol. The system is implemented using visual C++ each object in the agent architecture is implemented using C++ class. The work of Bouillot N, (2005) fast event ordering and perspective consistency in time sensitive distributed multi-player games emphasize usefulness of consistency model, as a means of ensuring synchronization. This also brings to light that consistency, synchronization contribute to enabled and effective mobility on distributed platforms.

## ENABLING AGENT MOBILITY ON GAME PLATFORMS

The distributed nature of game platforms as exposed in this investigation underpins and highlights the fact that mobile agents deployed on game platforms need to satisfy certain key characteristics in order for agents to exhibit mobility. This work provides new insights and directions necessary in capturing essential requirements when designing mobile agents for distributed system applications such as computer and internet games. According to our findings mobility platforms requirements can be classified into four main groups, these are; 1. **Timing requirements** - Latency (response times) and Synchronization 2. **Behavioural requirements** - Polymorphism, Inheritance, Persistency, Calling, Invocation, location, message passing; 3. **Addressing requirements** - Location, Naming and Encoding, 4. **Security requirements** - Availability, Self Protective, Fault tolerance and Certified Figure 1 summarizes generic requirements for distributed and mobility platforms. According to our study mobility



**Figure 1 - Generic Distributed Mobility Platform, Source: Melomey et al (2007)**

## GENETIC ALGORITHM

In this section we introduce genetic algorithm as a tool for formulating a fitness function for modeling mobility in mobile agents.

Genetic algorithm is a randomised search method based on the biological model of evolution through mating and mutation. This randomised search method is effective for constraint based problems. These problem solutions are encoded into bit strings that are tested for fitness; the best strings are combined to form new solutions using methods similar to the Darwinian process of survival of the fittest and exchange of DNA which occurs during mating in biological systems. Williams G, B (2007a).

Genetic algorithm is usually traced to John Holland. In his publication Holland J (1975) Holland describes the ability of simple representations (bit strings) to encode complicated structures and simple transformations which have enough power to improve such structures. He also showed that with the proper control structure, rapid improvements of bit strings could occur under certain transformations, so that a population of bit strings could be made to evolve as a population of animals would. One important result was that even in large and complicated search spaces, genetic algorithms would tend to converge on solutions that were globally optimal or nearly so Williams G B (2007a).

## FITNESS FUNCTION FOR MOBILE AGENT MOBILITY

“Fittest function is derived from the criteria specified for fitness. For example in the natural world of sports, sportsmen and women have to pass a fitness test in order to be selected for a tournament. In this same regard a program is considered fit if it meets a certain criteria designed to pass fitness and be selected. Such a criterion for program fitness could include, loosely coupling and highly cohesive of the individual modules, procedures that form the program. Fitness therefore can be represented using different program inputs. Searching for the fittest program is mainly based on probability of the fittest function in the population of a particular generation. Programs can either be selected or passed over after this process” Williams G B (2007a).

The GA based mobility function for mobile agents modelling represents a set of functional and non functional requirements as binary string structures. Fitness criteria matching the binary string structures will be considered fit to optimise the development of a mobile agent based system. We believe this because a measure of the degree of match between binary bit string representations of the key function and non functional requirements can be defined. This representation is also effective as different levels of complexity can still be introduced into the matching function.

Now to get the optimal result for mobile agents mobility the principles involved in building genetic algorithm are applied.

- All key functional and non functional requirements had their fitness initialised to zero
- Fitness function of the mobile agent is based on the similarity of key requirements
- A key requirements, is randomly selected
- A sample of key requirements of size  $\mu$  is selected from the repository of requirements without replacement
- The score of each mobile agent is compared against the selected requirement. The mobile agent with the highest score had its score added to its fitness value. Fitness of all other mobile agent remains unchanged.
- The mobile agents are returned to the mobile agent population with the process repeated a number of times.
- Based on the fitness computed, a GA simulation is carried out with crossover and mutation to evolve the mobile agent population through one generation of evolution.
- The process is then repeated from selection of key requirements till convergence in the mobile agent population.

In establishing whether optimal solution will be made up of mobile agents with specialist mobility or generalist mobility features and capabilities depends on the sample size  $\mu$  (the control parameter).

## STEPS FOR IMPLEMENTING FITNESS FUNCTION

Randomly create an initial population of mobile agents  $m(0)$

1. Compute the fitness function  $u(m)$  for each individual mobile agent  $m$  in current population  $m(t)$
2. Define probability for selection  $p(m)$  for each individual mobile agent in  $m(t)$ , such that the probability  $p(m)$  is equal to  $u(m)$
3. Generate  $m(t+1)$
4. Select individual mobile agents using probability  $m(t)$  producing new agents known as offspring via crossover, mutation or reproduction.

## MOBILE AGENT MOBILITY FITNESS FUNCTION

Let  $F$  be the function denoting key mobility requirements for a mobile agent.  $f1$  to  $f15$  are elements in the same set  $F$ .

$f1$  = Synchronization  
 $f2$  = Latency,  
 $f3$  = Abstraction  
 $f4$  = Polymorphism  
 $f5$  = Inheritance  
 $f6$  = Persistency  
 $f7$  = Calling  
 $f8$  = Invocation  
 $f9$  = Message Passing  
 $f10$  = Naming  
 $f11$  = Addressing  
 $f12$  = Encoding  
 $f13$  = Availability  
 $f14$  = Replication  
 $f15$  = Self Protective and Certified

$$F(X) = (x1.....xn)$$

The fitness function  $u(m) = (x1.....xn)$

Where  $u(m) = (1/e+x)^2$

The above expression represents a fitness function in an inverse relationship to a fitness solution.

The fitness solution derived from the fitness function is applied in the second of the four major phases thus;

1. Mobility requirement
2. Mobility analysis
3. Mobility design
4. Implementation of code.

See figure 2 for conceptual view of mobility model as part of the generic methodology.

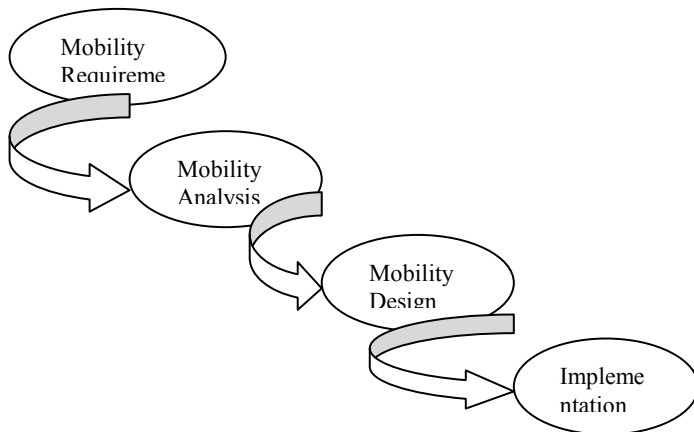


Figure 2: Mobility Model

## CONCLUSIONS & FUTURE WORK

In this work we classified and modeled mobility requirements using a fitness function derived from genetic algorithm. The criteria for formulating the function has been based on ; **Timing requirements** - Latency (response times) and Synchronization , **Behavioural requirements** - Polymorphism, Inheritance, Persistency, Calling, Invocation, location, message passing; **Addressing requirements** – Location, Naming and Encoding, **Security requirements** - Availability, Self Protective, Fault tolerance, Replication and Certification. The fitness function expressed will be exploited in more detail as a bench mark in determining whether a mobile agent satisfies key requirements for exhibiting mobility on a game platform generic requirements for distributed and mobility platforms.

## REFERENCES

Brustoloni, J.C. 1991. "Autonomous Agents: Characterization and Requirements." Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh: Carnegie Mellon University

Bancroft M. and D. Al-Dabass 2004. " A Combat Simulation Aid for Dungeon and Dragons". In *Proceedings of 5th Game-On International Conference*, pp 60-65. Reading,UK.

Cubaleska, B. and M. Schneider 2002 "Detecting DoS Attacks in Mobile Agent Systems and using Trust Policies for their Prevention" , Policy Workshop, International Workso on policies for Distributed Systems and Networks POLICY 2002: 198-201

Foster I., C. Kesselman , J.Nick , S.Tuecke 2002. "Grid Services for Distributed System Integration Computer." IEEE, 35(6).

Fukutake H., Y. Okada and K. Nijjima 2004. " 3D Visual Component based voice on input/output interfaces for interactive graphic applications." In *Proceedings of 5th Game-On International Conference*, pp 20-24.

Franklin S. and A. Graesser 1996. "it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, Institute for Intelligent systems University of Memphis.

Galli D.L. 1999. "*Distributed Operating Systems: Concepts and Practice*". Prentice Hall, 1<sup>st</sup> Edition

Gavrilovska A., S. Kumar., S.Sundaragopalan., and K. Schwan 2005. "Platform Overlays: Enabling In-Network Stream Processing in Large-scale Distributed Applications", In *Proceedings of 15<sup>th</sup> International Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV* (Skamania, Washington, Jun 13-15).*Communications of the ACM* : 158113987

Henning M. 1998 " Binding, Migration, and Scalability in CORBA." *Communications of the ACM*, Vol. 41, No 10 (Oct.)

Holland J, "Adaptation in Natural and Artificial Systems"; 1975

Jansen W. 2002. " Intrusion Detection with Mobile Agents". *Computer Communications, Special Issue on Intrusion Detection Systems*, vol. 25, num. 4, September.

Jennings N. R. 2000. "On Agent-Based Software Engineering" *Artificial Intelligence Journal* 117 (2) 277-296.

Jennings N. R., K. Sycara and M. Wooldridge 1998. "A Roadmap of Agent Research and Development" *International Journal of Autonomous Agents and Multi-Agent Systems* 1 (1) 7-38.

Melomey D, Williams G B, Imafidon C, Perryman R 2007. 11th International Conference on Computer GAMES: AI, Animation, Mobile, Educational and Serious Games

Milojicic D., D. Kotz, D. Lange , C. Petrie, C. Rygaard 1999. "Mobile agent applications" *IEEE Concurrency* (July to September) pp 80-90

Mockapetris, P., K. Dunlap 1988 "Development of the Domain Name System". *Communications of ACM SIGCOM*, Stanford, CA, pp. 123-133.

Neema S., J. Sztipanovits., G. Karsai. And K. Butts 2003. "Constraint-BasedDesign-Space Exploration and Model.

Synthesis.” *Proceedings of Third International Conference on Embedded Software (EMSOFT)* , LNCS 2855, pp. 290-305.

Overmars M. 2004. “Game Design in Education.” In *Proceedings of 5th Game-On International Conference*, pp 9-18. Reading,UK.

Postel, J., C. Anderson, 1994. “White Pages Meeting Report”, RFC 1588, (Feb.)

Roman, G.-C., G.P Picco, and A.L.Murphy, 2000. “Software Engineering for Mobility: A Roadmap,”In *Proceedings of 22nd International Conference on Software Engineering Future of Software Engineering*, A. Finkelstein (ed.), (invited paper) pp. 241-258.

Simatic M., S. Craipeau, A. Beugnard, S. Chabidon, M-C Legout, E. Gressier. “ Technical and Usage issues for Multiplayer games”. In *Proceedings of 5th Game-On International Conference*, pp 134-138. Reading,UK.

Smith, D. C., A. Cypher and J. Spohrer 1994, "KidSim: Programming Agents Without a Programming Language," *Communications of the ACM*, 37, 7, 55-67

Spence D., J. Crowcroft,S. Hand. and T.Harris 2005. “Location Based Placement of Whole Distributed Systems.” In *Proceedings of the ACM conference on Emerging networking experiments and technologies*. PP 124—134.

Thorn D., I. Palmer and E. Williams 2004. MMORG on Mobile Devices? Considerations When designing distributed

Adventure games”. In *Proceedings of 5th Game-On International Conference*, pp 150-154. Reading,UK.

Williams G.B 2000 “Technical Notes in RMI” Achival & Unpublished Work

Williams G.B 2007a. “Artificial Intelligence” Existing & Emering Techniques” Google 1<sup>st</sup> Edition

Williams G.B 2007b. “ Online Business Security Systems”. Spinger , 1<sup>st</sup> Edition.

Wright T. 2004. “Naming Services in Multi-Agent Systems: A Design for Agent-Based White Pages”. In *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems- AAMAS* pp 1478-1479

Yeo, A., Ananda, A., Koh, E. 1993 “A Taxonomy of Issues in Name Systems Design and Implementation”, *Communications of ACM SIGOPS Operating Systems Review*, New York, NY, July , pp. 4-18

Zeng X., Q.H. Mehdi and N.E. Gough 2004. “Implementation of VRML and Java for Story Visualisation Tasks”. In *Proceedings of 5th Game-On International Conference*, pp 112-126. Reading,UK.

# Modelling Mobile Agent Mobility in Virtual Learning Environment (VLE) using Fitness function

Divina Melomey, Godfried Williams, Chris Imafidon, Roy Perryman  
University of East London, School of Computing and Technology  
Docklands Campus, United Kingdom,  
E-mail: {divina,G.Williams,chris12, perryman }@uel.ac.uk

**Abstract:** Virtual learning Environments are driven by distributed systems. Effective distributed systems communications requires intelligent mobility as a vehicle to enabling seamless resource sharing and access to services. The nature of VLEs requires software tools for managing and making learning enjoyable and less painstaking. Mobile agents enable different software and services to collaborate in information sharing, adapt to new service requirements, demonstrate cooperation in a system environment, however being independent and autonomous. These requirements are essential in achieving mobility in VLEs. This work presents a novel fitness function as a key feature of a generic software methodology for modeling mobile agent mobility in VLEs.

## Introduction

Virtual Learning Environments (VLEs) are group of software used for managing and enhancing learning electronically (Roach & Stiles, 1998). This facilities and functionality enables tutors, instructors and students communicate online (Ginsburg, 1998). VLE should have the capabilities to enhance student learning experience based on the requirements of the programme a student is enrolled hence enriching students learning experience. Heaton-Shrestha, , Ediringha, Burke and Linsey (2005) also defined VLE as web-based software products providing sets of internet tools to enable teaching materials to be managed. Pablo and Wallace (2001) explained the VLE is not only dependent on the its accessibility, availability and the integration of the technology for the benefits of students but rather on the willingness of tutors to embrace and use computers for the delivery of course materials. Apart from VLE supporting teaching, learning and certain administrative functions, it also has the ability to facilitate communications among learners (Booth & Hulten 2003). The modes of communication are both asynchronous and synchronous. Again, VLE mode of delivery can be synchronous, asynchronous and/or both (Chen, Li & Shyu, 2003). These forms of communications are emails, booking appointments, negotiating assignment deadlines, social interactions with other students via blackboard learning.

This paper reports on a study conducted to ascertain the requirement for developing Virtual Learning Environments (VLEs) and how these needs are met using fitness function for modelling the solution to meet the requirements and demands of such as system. The systems used for this study was University of East London blackboard Learning System called UEL Plus. We realized that UEL Plus has multiple features to support teaching and learning. UEL Plus provides an improved communication, access to resources and advanced assessment capabilities. Our study focused fundamentally on the UEL Plus which part of VLE. The rest of the paper is organized as follows: Section 2 will describe end user categories and section 3 will highlight the mobile agents as a solution. Section 4 will introduce mobile agent fitness function and Section 5 will discusses the mobility in VLE in section 6 draws conclusions.

## 2.0 User Categories

We identified two main user groups for this study. They are front-end and back-end users. According to Sampson, Karagiannidis, Schenone and Cardinali (2002) formal, vocational, life long and occasional learners fall under the front end users category while individuals, software houses and other organization whose main interest are developing management learning and virtual learning software. Basic functions and or task on a VLEs are;

1. Authentication and authorization
2. Editing and saving personal settings
3. Navigation through the site
4. Using available communication tools
5. Building course content
6. Assessment
7. File Upload

Front end users need these basic functions to be user friendly and easily accessible. This form of interactions between users and the software is at the heart of e-learning development.



Back end users use information and input from the front end users to map up these functions of the front end users to the solution provided by back end users with respect to developing knowledge repositories and resources.

### 3.0 Mobile Agents an Alternative Solution

Our experiences in evaluating UEL Plus identified certain areas where an agent could be used in modeling interactions and communications during the systems development as we believe that it will considerably improve performance and front end user experiences of UEL Plus. The areas where we had feedback relating to front end user experiences were:

- Uploading of files
- Maintaining files and folders on VLE
- Using communication tools for creating asynchronous discussion, emails and chat
- Monitoring and tracking progress of students
- Other emerging technologies that could be added on

Based on these feedbacks, we proposed a solution into the modelling of mobility in mobile agent for VLEs. Gutl et al. (2004) identified three main objectives as an innovative solution in e-learning systems. These objectives were;

- Personalized retrieval of information,
- Presentation and management of relevant learning material in a timely fashion; ability to support teaching and learning paradigms and lastly
- An improvement on knowledge with respect to front end users behaviour in human to computer interaction.

In the following section we will show how we used the fitness function to model solutions for the critical areas of applications that require mobility such as VLEs.

### 4.0 MOBILITY FITNESS FUNCTION

Mobility fitness function is a function derived from an algorithm, based on the concept of survival of the fittest in genetics. In this section we defined elements for the mobility requirement for the mobile agent. The list is not exhaustive but only a representation for the fitness function.

Let  $F$  be the function denoting key mobility requirements for a mobile agent.

Let  $f_1$  to  $f_{15}$  be elements in the same set  $F$ .

$f_1$  = Synchronization

$f_2$  = Latency

$f_3$  = Abstraction

$f_4$  = Polymorphism

$f_5$  = Inheritance

$f_6$  = Persistency

$f_7$  = Calling

$f_8$  = Invocation

$f_9$  = Message Passing

$f_{10}$  = Naming

$f_{11}$  = Addressing

$f_{12}$  = Encoding

$f_{13}$  = Availability

$f_{14}$  = Replication

$f_{15}$  = Self Protective and Certified

Melomey, Williams, Imafidon, & Perryman (2008) established the implementation of generic mobility fitness function based on the following steps:

- Initial population should be randomly created for mobile agent  $m(0)$
- the fitness function  $U(m)$  should be computed for each individual mobile agent  $m$  in current population  $m(t)$
- probability for selection  $p(m)$  for each individual mobile agent in  $m(t)$  should be defined, such that the probability  $p(m)$  is equal to  $U(m)$
- $m(t + 1)$  generated

- Selection of individual mobile agents using probability  $m(t)$  to produce new agents which is known as offspring via crossover, mutation or reproduction

Let  $F(X) = (x_1, \dots, x_n)$

The fitness function  $U(m) = (x_1, \dots, x_n)$

Where  $U(m) = (1/e+x)^2$

$$U(m) = m(t) + \sum_{x=1}^n m(t+1) f(x)$$

The above expression represents a fitness function in an inverse relationship to a fitness solution.

The fitness solution derived from the fitness function is applied in the second of the four major phases thus;

1. Mobility requirement
2. Mobility analysis
3. Mobility design and
4. Implementation of code.

#### 4.1 Fitness Function for VLE

In the following subsection we will show how we used our fitness function to provide solution for VLE issues identified in section 2.

##### *Addressing*

There are certain elements that need to be present for an entity say agent to be able to travel from its platform of origin  $Hp_i$  to a host platform  $Vp_n$ . These requirements are required to perform address resolution prior to process migration. Three elements that need to be present are:

Receiver identification (RID)

Packet identification (PID)

Transmission Frequency of physical layer (TF)

Let R be the set requirement RID, PID, TF

Let H be the set header fields that contains control information

$L$  be length of the packet

$p$  be payload type

$s$  be sequence numbers

$i$  be integrity check information

$$R \subseteq H$$

Each computing platform is identified by global assigned address. A process will be able to migrate if it contains a header field carrying control information. The address resolution client which is the host platform needs to verify the integrity, authenticity and the logical address for resolving information sent across different platforms.

A platform hosting each mobile agent need to ensure mobile agents on its platform has a valid server and address resolution is also valid. Authorisation of available address to be used should be authorised by servers in order to ensure validity of the address.

##### *Replication*

High availability of services is paramount to mobile distributed computing as it enhances performance. It is a technique that is used to maintain copies of data in geographically dispersed environment and also as a back up in the event of loss of data or a systems failure (Coulouris, Dollimore, & Kindberg, 2005). The fitness of a replica will be measured in real time by the function of the differences in elapsed time. This ensures consistency and correctness at anytime for events. This represented as:

$$F(t): f_{t+1} - f_{t-1}$$

Where  $f_{t+1}$  is the current time replica server was accessed

$f_{t-1}$  last known time a replica was accessed

### *Remote Method Invocation (RMI)*

A method transparently invoked from process A to process B across a network as if it were a local method is termed as remote method invocation (Coulouris et al.2005; Williams 2000). This holds true for object oriented language rather than a procedural language. Invoking a method remotely involves two processes:

1. a reference to the remote object
2. a registry to store remote references

Let  $n$  be the number of identified elements for solution  $X$

$x_i$  be elements in  $X$

$f(x_i)$  the fitness of  $x_i$

The fitness of  $F$  can then be defined as

$$F(X) = \frac{1}{n} \sum_{i=1}^n f(x_i); n > 0$$

We define the average fitness above as average fitness for the elements in the mobility requirements as identified.

$$F(x): Hp_i \rightarrow Vp_n$$

### *Persistency*

The Object Management Group (OMG) service stipulates a typical structure for persistency. This should consist of persistent ID, persistent object, persistent object manager, persistent data store and protocol. A persistent object or entity that need to travel from Home platform ( $Hp_1$ ) to visit ( $n$ ) number of visiting platform ( $Vp_n$ ) require a reference ID, a dynamic state that lives the duration of the process and a persistent state that will be used for reconstruction of the dynamic state in case of a failure. These conditions qualified for an entity to be mobile in an environment.

### *Naming services*

The Sun Microsystem naming services system administration guide defines naming services as a central repository that computers, end users, and applications communicate together across the network. In this work, we also define name services as integrated services that manages all name information and hierarchies and also as an autonomous feature for transparency and persistency of entities (Melomey et al. 2007). Its function is to provide basic function and mapping of name to address on the network. In order to get the remote computer's address, the program must request assistance from say  $Hp_1$  from the domain name services (DNS) database running on that platform. DNS is a naming service which provides identification for computers on the internet. The name server uses  $Hp_1$  as part of the request to find IP address of the remote computer. The name server returns this IP address to the  $Hp_1$  only if the host name is in its database. It uses a logical tree to resolve names as part of the service

### *Synchronization*

Synchronization is important to maintain consistency of processes from  $H_p$  to  $V_{pn}$  at any given time (Coulouris et al.2005). The concept of clock synchronization deals with the understanding of ordering of events occurrence as produced by current processes. These events occur between message sender and message recipient for example from process A to process B. Clock synchronization is required to provide mechanism that can assign numbers sequentially based on agreement between sending and receiving processes. Several algorithms were developed over past decades. Lamport (1978) introduced the concept of an event happening before another in distributed environment. The notion is illustrated between event  $a$  and  $b$ ;  $a \rightarrow b$  where  $a$  "happens before"  $b$ . Another algorithm developed by Lamport and Meilliar-Smith (1985) require a reliable connected network to handle fault. Christian's algorithm measures in local time the time at which a message is sent ( $T_0$ ) and the time at which a message is received ( $T_1$ ). This is done by issuing a remote procedure call to a time server to obtain the time. The delay in the network is then estimated as  $(T_1 - T_0)/2$  (Christian, 1989). Hence the new time can be said to be the time returned by the server and in addition to time elapsed by the server to generate the timestamp. This is expressed by

$\text{Time}_{\text{new}} = \text{Time}_{\text{server}} + (T_1 - T_0)/2$ . There is also the Berkeley algorithm which was developed by Gusella and Zatti (1989). Berkeley algorithm was based on the assumption that any computer on the network has an

accurate time which can be used for synchronizing time between processes. This assumption may introduce delays and losses depending on the network and also due to the distributed nature in accessing the network and the processing capabilities on the learning system.

Let  $S$  = Synchronization

$H_p$  = visiting platform

$V_p$  = visiting platform

$V_{pn} = n$  visiting platform

$P_n = n$  number of processes

The timescale for measuring  $\Delta s$  is important where  $S$  which synchronisation is a derivative of the  $f(x)$  which is  $\Delta f / \Delta s$ . Measuring the short time for  $n$  processes is dependent on how fast changes occur in the system. The time range between which  $n$  process leaves  $H_p$  and arrives at  $V_{pn}$  can be expressed as:

$$F(x) \rightarrow \Delta t = \int_t^{t+\Delta t} f(s) dt \quad \text{where the interval is } [t, t+\Delta t]$$

$F(x)$  is a complex system during its evolution; the system may change its own  $F$

## 5.0 Discussion

In our study using UEL Plus, we analyzed feedback, identified student lecturer issues and evaluated mobility solutions for back-end user category. Solutions we designed using mobile agent oriented approach addressed synchronization, remote method invocation, addressing and naming services, persistency and replication of data. We examined the persistency of data and how they were mapped into the objects. We enabled the mobile entity to have an internal mechanism which acts as a persistency layer such that it will encapsulate database access from other objects. In this manner, data persist after any form of interruption and interaction occurs during the course. A fitness function for modelling and testing features appropriate for persistency of objects is critical in such as environment.

Front end users are more interested in up to date, timely and current state of databases. This implies that concurrent data access and update of repositories should be synchronized. This is more crucial when it comes to coursework submission for group projects, where continuous and joints updates are required from individual team members when approaching deadlines. Synchronization then becomes an issue for the back end users to deal with in order to ensure consistency of data, processes and clock synchronization of various remote devices connected to the network infrastructure. Our work indicates that there is a connection between replication of data at various server locations with respect to change in time among primary and secondary servers. This also applies to resolution of names and addresses.

We had the understanding that front-end users were looking for a unified point of authentication for ensuring coherent and an organized teaching and learning resource platform. Consistent and coordinated naming of objects and identification of processes underpins the need for metadata as a means of providing effective mobility. These needs are met based on the conditions that must be met for remote method or data invocation's fitness function criteria. The fitness function measures the suitability for elements mobility in the VLE.

## 6.0 Conclusion

In this paper, we presented an overview of VLE and user categorization. We also presented fitness function for mobility as alternative solution to traditional approaches in eliciting requirements for implementing mobility in VLEs. This mobility fitness function was further illustrated by applying it for mobility element requirements specification. This was further narrowed down to individual mobility requirement mapped unto their fitness solution applicable to the development of VLE and it was used to provide a solution tailored for simulating effective mobility in UEL Plus.

Currently, work is being done to integrate this fitness function as part of a generic methodology for capturing mobility in mobile agent based systems and applications. This when concluded will provide a standard methodology for building applications where mobile agents are seen as an alternative approach to information systems development.

## References

- Booth, S., & Hulten, M. (2003). Opening dimension of variation: An empirical study of learning in a web-based discussion. *Instructional Science* Vol.36 (No.1&2), 65-86.
- Chen, S.-C., Li, S.-T., & Shyu, M.-L. (2003). Model-Based System Development for Asynchronous Distance Learning. *International Journal of Distance Education Technologies*, Vol.1 (No. 4), 39-54.
- Christian, F. (1989). A Probabilistic Approach to Distributed Clock Synchronization. *Distributed Computing* Vol. 3, 146-158
- Coulouris, G., Dollimore, J., & Kindberg, T. (2005). *Distributed Systems, Concepts and Design* (4th ed.): Addison-Wesley Publishers.
- Ginsburg, L. (1998). In *Technology, Basic Skills, and Adult Education: Getting Ready and Moving Forward*. Information Series (No. 372).
- Gusella, R., Zatti, S. (1989). The Accuracy of Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD. *IEEE on Software Engineering*, VI.15 (No.7).
- Gütl, C., Pivec, M., Trummer, C., García-Barrios, V. M., Mödritscher, F., (2005). AdeLE (Adaptive e-Learning with Eye-Tracking): Theoretical Background, System Architecture and Application Scenarios. *European Journal of Open, Distance and E-Learning (EURODL)*(2005/II).
- Heaton-Shrestha, C., Ediringha, P., Burke, L., & Linsey, T. (2005). Introducing a VLE into campus-based undergraduate teaching: Staff Perspectives on its impact on teaching. *International Journal of Educational Research* Vol. 43(6), 670-386.
- Lamport, L (1978). Time, Clock and Ordering of Events in Distributed Systems. *Communications of ACM* Vol.21(No. 7).
- Lamport, L & Melliar-Smith, P. M.(1985). Synchronizing Clocks in the Presence of Faults. *Journal of ACM* Vol.32(No. 1), 52-78.
- Melomey, D., Williams, G., Imafidon, C., & Perryman, R. (2008). *A Fitness Function for Capturing Mobile Agent Mobility on Games Platform* Paper presented at the 12th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia & Serious Games. , Louisville, Kentucky, USA.
- Melomey, D., Williams, G., & Imafidon, C. (2007). *Mobility Requirements on Game Platforms: An Agent Perspective* Paper presented at the 11th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia & Serious Games. , University of La Rochelle, La Rochelle, France.
- Object Management Group (2000). *Persistence Object Service Stand-alone document*.
- Pajo, K., & Wallace, C. (2001). Barriers to the uptake of web-based technology by university teachers. *Journal of Distance Education* Vol. 16(No.1), 70-84.
- Roach, M. P., & Stiles, M. J (1998). COSE - A Virtual Learning Environment founded on a Holistic Pedagogic Approach. *CTI: Software for Engineering Education* (No.14).
- Sampson, D., Karagiannidis, C., Andrea, S., & Fabrizio, C. (2002). Knowledge-on-Demand in e-Learning and e-Working Settings. *Educational Technology and Society*, Vol.5 (No. 2), 107-112.
- Sun Microsystems, Inc. (2003). *System Administration Guide: Naming and Directory Services (DNS, NIS, LDAP)*.
- Williams, G.B. (2000). *Technical Notes in RMI*. Archival and Unpublished.

# MOBILITY CHALLENGES IN ONLINE APPLICATION DEVELOPMENT

**Divina A. Melomey**

The need for mobilization and mobility in online applications has become paramount in ensuring that client and customers' needs are met at the point of service. This work validates detail mobility requirements for such applications. This validation covers areas such as e-health, online banking and virtual learning environment platforms. Further investigations with regards to these requirements explored mobility in games. Initial work to date has highlighted generic mobility requirements derived from distributed platforms. These requirements implementation validates parameters such as persistency, concurrency, message passing and calling, synchronisation, transparency, serialization and remote invocation in the application environment mentioned. This talk explores requirements needed for development of online applications and how mobility requirements are met. These requirements are then validated against existing online applications with case analysis. This validation process is crucial for assessing mobility of online applications for clients or customers.

Divina A. Melomey

School of Computing and Technology, Docklands Campus, University of East London,  
United Kingdom

`divina@uel.ac.uk`

# Implementing e-learning and Web 2.0 innovation

## Didactical scenarios and practical implications

**David Durkee, Stephen Brant, Pete Nevin, Annette Odell, Godfried Williams, Divina Melomey, Hedley Roberts, Chris Imafidon, Roy Perryman and Anna Lopes**

**Abstract:** *This paper examines the practical implications for teachers wishing to incorporate e-learning and Web 2.0 technologies into their pedagogy. The authors concentrate on applied didactical scenarios and the impacts of e-learning innovations. The methods applied stem from grounded theory and action research. An analytical framework was derived by inverting problem-based learning (PBL). Three practices at the University of East London (UEL) are examined in the context of this framework, using, respectively, a formal virtual learning environment, Facebook and Skype. The paper's findings have implications and provide guidance for those planning and implementing online collaboration and learning in education and industry.*

**Keywords:** *e-learning; Web 2.0; communities of practice; Facebook; Skype; VLE*

*The authors are with the University of East London. David Durkee is in the Department of International Development: 3rd World, School of Social Sciences, Media and Cultural Studies (SSMCS). Stephen Brant is with UELconnect. Pete Nevin and Hedley Roberts are in the School of Architecture and Visual Arts (AVA). Godfried Williams, Divina Morley, Chris Imafidon and Roy Perryman are with the School of Computing, Information Technology and Engineering (CITE). Anna Lopes is in the Department of Anthropology, SSMCS. Contact: David Durkee, International Development: 3rd World Programme, University of East London, 2–4 University Way, London E16 2RD, UK. E-mail: durkee@uel.ac.uk.*

Overcoming barriers to lifelong learning requires innovation in practice to keep pace with technological and social realities. Where computing has become more ubiquitous, some groups have managed to successfully implement e-learning and Web 2.0 technologies. While qualified innovation and best practice prove essential elements in advancing the quality and delivery of learning, as Valcke and DeWever (2006, p 40) observe, few studies define the precise role of information

technologies (IT) in education (p 40). To that end, in this paper we describe a series of didactical scenarios with an applied role for IT.

In many contexts, a virtual learning environment (VLE) successfully acts as a one-stop-shop for students' online study needs. In theory, a university's VLE can be used to facilitate the professional development of students. However, various inhibiting factors appear to limit the potential of the VLE in several learning and

teaching scenarios. First, the VLE does not easily allow anyone but those in the development and delivery team to upload information or resources. Although the concept of Web 2.0 has been prevalent in recent years, whereby students and teachers alike can construct knowledge and meaning together, only a few UK universities have effectively implemented their VLEs in this manner. This means that in most cases students do not have a means of disseminating knowledge and information among themselves or of contributing to the bank of learning resources. Second, most VLEs are closed internal systems that limit access to students and teachers who are registered at the institute. Many courses require students to produce work in various media. The ability to share work-in-progress or examples of completed work (either with tutors or with other students) is therefore something that technology needs to facilitate. In addition, students' future employers will require a facility with technology as an employability skill: the development of professional skills is thus partially achieved through the integrated experience of online publishing. A VLE obviously requires intentional log-in: publishing within a VLE consequently restricts the number of people who may see what has been published. Third, there is a problem with the frequency of student log-ins to VLEs. With a VLE, or even a university e-mail account, students may log in once a week, or even just once a semester, if at all. Thus using the VLE as a communication tool may miss the mark, because students may not get information at the right time. Obviously, it can be argued that students, as part of their education, should be able to choose when and where they spend their time. Moreover, if they want the resources and they are available, surely they will get them. However, the question remains, where are the students and what are they doing?

In this paper, we assess how teachers can implement IT tools effectively to allow students to work and learn together actively as part of their preparation for their future careers. Two of the three practices described in this paper adopt IT tools frequently used by students (*Facebook* and *Skype*) to complement the classroom experience. In our analysis, the most pervasive didactic scenarios have one of two key features: (a) the connection of remote students to resources (whether course materials, a teacher or even an event to facilitate participation); and (b) the involvement of a remote expert. The first scenario is characterized by the fact that the students are studying outside the traditional classroom and are enabled to connect to learning resources. Elsewhere in this special issue of *Industry and Higher Education*, examples are provided of this scenario in the cases of prospective students who follow

an online preparatory course before the start of their academic programme using Web videoconferencing (Giesbers *et al*, 2009), professionals who undertake part-time study using *Wikis* (McLuckie *et al*, 2009), and a learning programme designed to enhance analytical skills using discussion forums (Rehm, 2009).

Conversely, the second scenario has the students in the classroom and the resources (the expert, a CEO, research papers) at a remote location. Here, the tutor may use the Internet to connect to the resources. However, if they are sitting at computers in the lesson, the students may also access those resources online. The defining characteristic of the second scenario is the remoteness of the resources, while the students and teacher are often in the same location, such as a classroom. This scenario seems to be often considered but less often implemented, and yet it affords a significant opportunity to enhance educational delivery through the use of remote resources, which may be an expert on a live feed, or podcasts and online videos. Thus the two generic scenarios are differentiated by the physical presence or non-physical presence of resources, students and teachers.

## Methods

In this paper, we apply grounded theory (Dick, 2005), a research methodology based on action research. An investigation of best practices at the University of East London (UEL) identified various factors that could be incorporated into an analytical framework based on an inversion of problem-based learning. These factors related to cognitive organization/framing, authentic problems, student autonomy/team choice and common interests, prior knowledge/misconceptions, and teacher support/demonstrations. Taking these factors into account, we were able to calibrate the range of best practices and practical innovation in evidence. In essence, the range extended from the use of the official virtual learning environment, UELPlus, to the use of social networks such as *Facebook* and *Skype*.

In the case of *Facebook*, students at UEL's School of Architecture and Visual Arts (AVA) used the site in cooperation with the teacher. Three advantages of using *Facebook* in class are:

- it allows a more varied group to become involved, including those normally 'locked out' of a VLE because they have no password – such as alumni, practitioners and potential employers;
- it allows students to 'publish' work both for critical appraisal and as part of their professional development, providing input for external examiners; and



- both tutors and students can post information on relevant events (such as exhibitions).

In the case of *Skype*, a presenter (an external expert) had to be at a conference in Geneva while teaching at UEL (Durkee and Brant, 2008). By using *Skype* in a live learning laboratory setting, the presenter could be physically present at Geneva and at the same time teach his class online using synchronous communication tools. In other words, through Internet technologies for sending live audio and video, the teacher was able to participate in person.

In the VLE case, we concentrated on the technical aspects relating to the architecture of virtual learning environments – in particular on areas of innovation related to mobility and trust. Our approach adopts the frequently-used Community of Inquiry model proposed by Garrison *et al* (2001) as a template and tool. This distinguishes three elements: ‘cognitive presence’, ‘social presence’ and ‘teaching presence’. To these we add a fourth, ‘technical barriers’. One should note that the priority here is not the use of *Facebook* or more generally Web 2.0 technology, but rather the selection of the technology that will best foster a community of practice. This point applies regardless of whether the didactical scenario features remote students or remote resources. If the right technology is chosen for interactions among participants, students will become engaged in a community of practice, and the opportunities for learning in that community appear to increase as a function of the number of its members.

## Vignettes

### *Cognitive presence*

Our work shows that best practice with regard to cognitive presence must recognize the degree to which participants are able to ‘construct meaning through sustained communication’ (Garrison *et al*, 2001). The ability to construct meaning from interaction may be easier for those in the classroom, who are working together in a real-time environment. We found that blended learning environments provided an excellent means of improving cognitive certainty and maximizing the benefits of technology for e-learning applications.

*The Skype case.* Bromme *et al* (2005) observe that it is more difficult to establish common ground in an asynchronous than in a synchronous activity. How, then, can lifelong learners and their teachers learn together effectively when separated geographically? Using *Skype* or *Instant Messenger* may help learners to communicate effectively with the instructor despite geographical distance. Most assume that, as a communication tool, *Skype* can provide a synchronicity for discourse.

However, because of the way *Skype* was used in the UEL setting, students could not immediately react when they had a question, as they would have been able to do in a one-to-one conversation. This raises an important point for those interested in connecting to remote resources: a remote lecturer or presentation is for the most part asynchronous for the students. While the video feed may be live, and happening in real time, if students cannot ask their questions as and when they occur to them, the communication becomes asynchronous. Thus, in the case of the remote presentation using *Skype*, this dynamic was lost to a certain extent. Where the speaker normally cues the audience, in this case the audience actually became increasingly silent due to the affective pauses. The live learning lab achieved more than simple technology demonstration or offering a remote presentation: the audience was able to experience the physical and psychological reality of this form of learning. For example, the room was very quiet until the time came for questions, and the participants could thus understand how such an attentive silence imparts a certain degree of psychological stress that is seldom discussed when considering technologies for distance applications. In this way, the students experienced a visual example of the possibilities and limits of IT for knowledge acquisition in distance learning environments.

We conclude, like Bromme *et al*, (2005, p 95), that paralinguistics (intonation, pitch, hesitation, gesture, etc), missing in such a learning environment, are very important in establishing an understanding and promoting interaction between audience and presenter. Thus when auditory yes/no cues or such gestures as nods are absent, the presenter must be more careful in designing the structure of the communication; otherwise audience response will be lacking. It is important to concentrate on enabling alternative means of interaction – such as circulating a wireless keyboard for typing questions.

*The Facebook case.* The use of Web 2.0 technologies, and social networking sites in particular, originated at UEL from feedback in one of its annual Student Satisfaction Surveys. The point was made in the 2007 survey that communication systems could be improved to effect, in particular, a rapid dissemination of information in the AVA community. Many courses in AVA require students to produce work in various media, and the ability to share work-in-progress or examples of completed work (with either tutors or other students) is therefore something that technology needs to facilitate. In addition, a facility with technology is an important employability skill in this field. Professional skills development is therefore partly achieved through

integrated experiences of online publishing (as in the case study below). The decision to experiment with social networking software (SNS), and with *Facebook* in particular, was partly based on the fact that many students are familiar with the software and frequently log on to it anyway (Sclater, 2008). This was confirmed by research done at the School. Of 226 students who completed a survey about technologies with which they were familiar before joining AVA, 70 knew about VLEs, but only 31 had used one. On the other hand, 202 knew about *Facebook* and 165 were currently using it. All 37 respondents studying for a BA in Graphic Design, for example, knew about *Facebook*, and 35 were already using it. The need for support in developing expertise in an unfamiliar technology was therefore considerably reduced and the technology itself was not a barrier to easy communication.

Interestingly, the use of *Facebook* offers the tutor an opportunity for valuable synchronous communication, because he or she is meeting the students in 'their' environment. It could be argued that this teaching approach mimics the patterns of students' typical interaction, and provides a certain type of auto-synchrony to the discourse that might be absent when a presenter comes in live on a *Skype* feed. The synchrony results from the act of being present. Students can see who else is there, and can turn to a friend or ask a question of the tutor as and when they need. However, because many of the current tools in the environment limit the type of interaction to text, a certain amount of asynchrony cannot be avoided. Nonetheless, one tutor noted:

'*Facebook* has been a revelation. I believe it certainly wouldn't be possible to achieve the same result in any other way. It would take so much administration and labour, but with *Facebook* the administration is minimal. Integration is a key word. I can organize all the blogs they have set up around the group, I can post information or resources, I can chat at any time and deal with possible problems in a direct and human way.'

Furthermore, since this is a student environment, the participating tutor appears more as a peer and less as the expert.

#### *Social presence*

According to Tu and McIsaac (2002), social presence is the interplay of two variables: intimacy and immediacy. They contend that 'intimacy' may be established through haptics, such as eye contact and body posturing. If something reduces the comfort level, argue Tu and McIsaac, people will change their behaviour to return to

#### **Student A**

October 16 at 4:08pm

hi pete, ill be in next week .. im getting a taxi paid for by the uni to take me .. im on crutches, waiting for an operation on my knee as ive torn the muscles in it and possibly have mini fractures in it...but ill be in. i heard everyones in groups now.. would that be a problem for me?

see you tuesday !

A

#### **Pete Nevin**

October 16 at 4:45pm

No problem. There are a few people that have not been in for various reasons so lets see at the time and maybe you can join a group or maybe Ill start a new one. Pete.

#### **Student M**

October 10 at 7:04pm

Hi pete ive just found out ive got my next set of infusions on tuesday so i wont be able to make it in again!! Not to happy about that .. so would you be able to give me some work for me to do in the hospital that day or before then, any drawings or what is the brief?

thanks and sorry again

M

#### **Pete Nevin**

October 10 at 8:46pm

Hi M

We have a project to design and make a drawing machine over the next two weeks. You could design and invent a fantastic machine like those that Heath Robinson designed. You should think about the machine like something you buy at IKEA with all the diagrams of construction. Use your imagination. And don't worry.

Pete

**Figure 1.** Communications between tutor and students in the *Facebook* case.

an 'optimal comfort level' so that the interaction can continue. 'Immediacy', they explain, refers not to the physical but to the psychological perception of closeness that results from verbal and non-verbal cues.

It was found that immediacy relates directly to the perception of the tutor's effectiveness. Where the tutor is present and available, the students respond in kind. Similarly, frustration and other negative sentiments concerning on the tutor's efficacy are closely associated with a lack of immediacy (Tu and McIsaac, 2002). In terms of best practice, therefore, the goal is to select those technologies that provide an opportunity for the tutor to be present without needing to be active all the time. It is important to bear in mind that it is the perception of immediacy, and not its actuality, that yields the effect. Thus a good technological interface can improve social presence through immediacy.

*The Facebook case.* Figure 1 presents extracts from communications between a tutor and two of his students. The extracts illustrate how social interaction occurred using *Facebook*. By reacting quickly to the

concerns of the students, the tutor established social presence. In the second extract, by integrating social and teaching elements, the tutor tried to help the student to continue working on his assignment despite his personal problems.

#### *Teacher presence*

De Laat *et al* (2006) examine the nature of online teaching in a networked learning community. They focus on how Internet technologies can promote collaboration and cooperative connections in at least three dimensions: 'learner → learner'; 'learner → teacher'; and 'community → resources'. Their research points to changing roles for both teachers and students. In their vision, the teacher becomes a guide on the side (p 2). Students assign roles and schedule activities instead of simply exploring or regurgitating content to complete a task. Since it has already been shown that the teacher must engineer the environment to get the most from it, even if this is simply considering where the resources are located and how the students will access them, there will still be a teacher presence. Goodyear (2002) found that 'teaching online requires new skills for the teacher as well as a different attitude towards teaching or being a teacher'. One of the greatest challenges for the teacher is to demonstrate the relevance of the media (Goodyear, 2002, p 4). Blended learning helps greatly with this. Mixed learning overcomes the online danger of miscommunication, as people can discuss points face to face on a regular basis. In this approach, the online communication becomes a continuation and an extension of a discourse that is already underway. Moreover, since it supports and supplements the discourse, rather than taking responsibility for the primary dialogue, the online dynamic can be much freer.

*The Skype case.* In the *Skype* case, the teacher played a dominant role as he was directly instructing his class at the beginning of the session. The experience of the live learning lab confirmed that two key factors to consider in facilitating equality and fairness are the role of the technology and the method allowed for interventions. Had a keyboard been used at the end of the presentation, the question and answer session might have been more substantial. While less can be done in this environment in terms of looking at an individual or being able to hear those who are talking at the back of class, the teacher must take an active role in making the learning environment one in which students can make themselves heard and can feel at ease to practise what they are learning.

*The Facebook case.* A tutor may be perceived differently in a social networking context and therefore

#### **X's Wordpress Weblog entry:**

I am unsure what the issues are on music about the US Laws as on One True Media, they have this link to music where you can add these samples (like in this movie). As this is just a first edit, there is no writing yet. ...

#### *Facebook communication:*

##### **Student Y**

October 16 at 11:57pm

Hi Pete.. I am concerned as I did not hear the conversation concerning the Laws on music on Youtube. I have had many troubles over Jump cuts, so have been frantically figuring out One True Media. I then had my own WMA sound files, but as I cannot (and will try to convert to) MP3 format, this was unsuccessful. So then I saw they had music on a menu, and had these samplers... Do you know (or will I have to research this?) as to whether this would infringe the copyright for Youtube? Any advice will be much appreciated

Y

P.s, could you please give any advice to my movie to help it along in any way... (the harsher the better). Thanks.

[url to the work on her blog given]

Take care...

##### **Pete Nevin**

October 18 at 12:37pm

You will have to try and get your sound files to work. Your own sound is what you need. The copyright laws are the same everywhere so don't use the work of others. If the site provides samples then I would imagine you can use them if it's open source media. Pete

##### **Student X**

October 18 at 12:57pm

cheers pete...yes, not happy with the text either. Have just redone the text, so just about to see if those have worked better. The music worked better when there was no text, now there is, I hate the music!!!

Thanks for your help.

see you monday

**Figure 2.** Communication between students and tutor on copyright using *Facebook*.

needs to manage his or her presence carefully. From the extracts in Figure 1, it is clear that the tutor reacted in an open and constructive manner, as recommended by De Laat *et al* (2006) and Anderson *et al* (2001). He also reacted promptly and adopted a coaching style. This approach is also evident in the extracts presented in Figure 2 – in this case, a student is concerned about copyright laws.

Figures 1 and 2 provide examples of how, using *Facebook*, students take part in a community of practice: they become part of something bigger than just their class or their specific university studies. Their actions and interactions on the site help to nurture an approach and attitude conducive to lifelong learning. They may interact with their fellow students or with people with whom they will be working. The examples show that presence need not be conceptualized only for the present course. Rather, tutors should think of the

personal development of their students and aim to facilitate their transition from learner to practitioner.

#### Technical barriers

*The VLE case.* Valcke and DeWever (2006), by concentrating on the process, move the discussion from *whether* to use IT to *what type* of IT should be used. In so doing, they effect a shift in the focus of analysis and investigation. The basic functions of VLEs are: authentication and authorization; editing and saving personal settings; navigation through the site. The 'fitness function' approach applied to the VLE controls the amount of complexity for the front-end user. The fitness variable can be adjusted to lower the cognitive entry barrier for new students or to allow for more devices to be used for accessing the VLE. Thus the fitness function pre-selects the most basic level of skills that the students will need to operate and use the system. Thus, while new students may or may not have the computer skills required to use the VLE when they arrive at the university, the considerable simplification produced by the fitness function means that most will find little difficulty in accessing learning materials from the outset. That simplification also decreases the number of students who will need a remedial course before they can make use of the system.

However, it is important to note that the fitness function, in its simplification role, does not guarantee that all of the system's functionality will be accessible. Given the existence of the function, tutors and administrators may expect that their students will be able to use the VLE with no problems and access all the materials presented there, but this is not the case – there may be a considerable skills gap in some students which will leave them unable to use the system to its full potential. As a point of best practice, therefore, activities should be built into first-year modules to introduce students to the specific IT skills.

Front-end users expect databases to be up-to-date. Data access and the update of repositories should therefore be synchronized. This is especially important with regard to coursework submission for group projects, in which continuous and joint updates are required from individual team members when approaching deadlines. Synchronization then becomes an issue for the back-end users, who must ensure consistency of data, processes and the clock synchronization of various remote devices connected to the network infrastructure.

We noted that front-end users were looking for a unified point of authentication for ensuring coherence and an organized teaching and learning resource platform. Consistent and coordinated naming of objects and identification of processes underpin the need for

metadata as a means of providing effective mobility. Currently, work is being done to integrate the fitness function as part of a generic methodology for capturing mobility in mobile agent based systems and applications. This will provide a standard methodology for building applications in which mobile agents are an alternative approach to information systems development.

As a point of best practice, our findings underscore the need for a single sign-on or point of entry, as this optimizes added-value transactions and ensures friendliness in a VLE. The importance attributed to user credentials cannot be underestimated, and the idea of federation may be extended by including service providers as well as end users. There is a strong case for service providers to have to submit credentials for verification and mutual authentication. Multi-factor authentication using a federated approach addresses key concerns such as user friendliness, usability, access to data and concurrency issues, performance, and migration and mobility among e-learners.

*Facebook case.* An alternative approach, as applied in the *Facebook* case, involves a system selected according to the designers' perceptions of what communication methods the students are using. By choosing a system which most students are already using, technical barriers can be diminished significantly. There are no problems with availability if *Facebook* is used: it just runs and is available to most students from virtually anywhere (especially now that it has a mobile version which makes it easily accessible from such devices as mobile phones). If a system is used in which students simply are present, pushing information is no longer an issue.

## Discussion

The application of technology in teaching can significantly enhance learning and understanding. However, the use of IT tools is circumscribed by the inherent benefits, limits and even risks of each technology. Technology can thus be both enabler and limiter at the same time. Furthermore, while many educators understand this paradox in theory, their practical experience is lacking. Moreover, little has been published with regard to practical didactical scenarios. A key factor underpinning the success of using IT to support learners is the role of the technology in promoting learning and in the delivery of content. Both the *Skype* and *Facebook* cases highlight the value of participatory learning. The audience and the students learn through the interaction. Thus it is important to underscore the significance of participation as a learning device. The participants become practitioners through presence.



There is a more generic problem with regard to cognitive presence and the quality of the information available. In an environment like the Internet, in which anyone can publish, anyone can be an expert. This means that searching for reliable information can be like looking for a needle in a haystack: while true expert information is available, it needs to be sought out and distinguished from the mass of less expert information. There can also be problem of too much information available, leaving the searcher at or beyond saturation point. The sense of overload can be compounded by information that lacks structure or organization. In fact, all these potential problems can occur when using a system like *Facebook* for pedagogical practice. The environment can become too 'rich'.

In the future, it would interesting to code the student–student, teacher–student and student–teacher dialogues using content analysis (Rourke *et al*, 2001) to assess the length and type of explanation. This would help us to ascertain whether there were any assumed *a priori* differences in the knowledge taken for granted or the style of communication. Such analysis could be especially helpful in understanding the role of community membership and in answering the question of Bromme *et al* (2005, p 98), 'What change does the expert make to his explanations in order to adapt them to the recipient's knowledge?'

## Conclusions

Little has been done in the field of distance learning and e-learning with regard to juxtaposing practice and research. A tendency appears in the literature for research to advance and inform. Continuing down that path didactically will shape the future known with the present known. In other words, the system will produce more and more of the same, albeit perhaps better validated. Studies are designed and teaching is observed. The 'model of community inquiry' proposed by Garrison *et al* (2001), with our addition of technical barriers, has proved helpful in identifying points of best practice. The fact that it could be applied to three very different projects is particularly relevant and demonstrates clearly the importance of presence when considering the practical implications of e-learning and Web 2.0 technologies.

As has been argued, a critical factor is the choice of an appropriate technology, as it this will directly influence all forms of presence. Where it is not possible to choose, then it is important to consider the design with great care. How can the delivery be engineered to provide the least degree of non-cognitive uncertainty and the highest level of immediacy?

With regard to the teachers, even if their role has changed, it is still vital that they are present and that

they ensure equality and fairness in the learning environment. The greatest technical barrier is not making use of the technology. When correctly applied, the technology can overcome many barriers and enable a paradigm of learning anytime and anywhere, with substantial benefits to learners.

## References

- Anderson, T., Rourke, L., Garrison, D., and Archer, W. (2001), 'Assessing teaching presence in computer conferencing', *JALN*, Vol 5, No 2, pp 1–17.
- Bromme, R., Hesse, F., and Spada, H., eds (2005), *Barriers and Biases in Computer-Mediated Knowledge Communication and How They May Be Overcome*, Computer-Supported Collaborative Learning Series, Vol 5, Springer, New York.
- Bromme, R., Jucks, R., and Runde, A. (2005), 'Barriers and biases in computer-mediated expert–layperson communication: an overview and insights into the field of medical advice', in Bromme, R., Hesse, F., and Spada, H., eds (2005), *Barriers and Biases in Computer-Mediated Knowledge Communication and How They May Be Overcome*, Computer-Supported Collaborative Learning Series, Vol 5, Springer, New York, pp 89–118.
- De Laat, M., Lally, V., Lipponen, L., and Simons, R. (2007), 'Online teaching in networked learning communities: a multi-method approach to studying the role of the teacher', *Instructional Science*, Vol 35, No 3, pp 257–286.
- Dick, B. (2005), 'Grounded theory: a thumbnail sketch', Version 1.07w, Resource Papers in Action Research, [www.scu.edu.au/schools/gcm/ar/arp/grounded.html](http://www.scu.edu.au/schools/gcm/ar/arp/grounded.html), accessed 18 March 2009.
- Durkee, D., and Brant, S. (2008), 'Enhancing teaching through technology: challenges and possibilities', Live Learning Laboratory UEL Learning and Teaching Conference, London, 2008.
- Garrison, D., Anderson, T., and Archer, W. (2001), 'Critical inquiry in a text-based environment: computer conferencing in higher education', *The Internet and Higher Education*, Vol 2, No 2–3, pp 87–105.
- Giesbers, B., Rienties, B., Gijssels, W.H., Segers, M., and Tempelaar, D.T. (2009), 'Social presence, Web videoconferencing and learning in virtual teams', *Industry and Higher Education*, Vol 23, No 4, pp 301–309.
- Goodyear, P. (2002), 'Online learning and teaching in the arts and humanities: reflecting on purposes and design', in Chambers, E.A., and Lack, K., eds (2002), *Online Conferencing in the Arts and Humanities*, Institute of Educational Technology, Open University, Milton Keynes, pp 1–15.
- McLuckie, J., Naulty, M., Luchoomun, D., and Wahl, H. (2009), 'Scottish and Austrian perspectives of delivering a Master's: from paper to virtual and from individual to collaborative', *Industry and Higher Education*, Vol 23, No 4, pp 311–318.
- Rehm, M. (2009), 'Unified learning – separated by space: case study of a global learning programme', *Industry and Higher Education*, Vol 23, No 4, pp 331–341.
- Rourke, L., Anderson, T., Garrison, D., and Archer, W. (2001), 'Methodological issues in the content analysis of computer conference transcripts', *International Journal of Artificial Intelligence in Education*, [http://ai.ed.ac.uk/members01/archive/vol\\_12/rourke/full.html](http://ai.ed.ac.uk/members01/archive/vol_12/rourke/full.html).
- Sclater, N. (2008), 'Web 2.0, personal learning environments, and the future of learning management systems', Research Bulletin No 13, EDUCAUSE Center for Applied Research Boulder, CO, [www.educause.edu/ecar](http://www.educause.edu/ecar).
- Schober, M.F., and Clark, H.H. (1989), 'Understanding by addressees and overhearers', *Cognitive Psychology*, Vol 21, pp 211–232.

Tu, C., and McIsaac, M. (2002), 'The relationship of social presence and interaction in online classes', *American Journal of Distance Education*, Vol 16, No 3, pp 131–150.

Valcke, M., and De Wever, B. (2006), 'Information and communication technologies in higher education: evidence-based practices in medical education', *Medical Teacher*, No 28, pp 40–48.