



UNIVERSITY OF NAIROBI

SCHOOL OF COMPUTING AND INFORMATICS

M.SC. IN INFORMATION SYSTEMS

AB3S: AGENTS BASED SERVICE SUPPORT SYSTEM

BY

NJUGUNA, PATRICK WANJIKU

P/56/P/8220/01

SUPERVISOR: MR .ELISHA OPIYO

A Project report submitted in partial fulfillment of the regulations governing the award of the degree of M. Sc. in Information Systems, at the University of Nairobi.

July 2010

University of NAIROBI Library




0439193 4

Declaration

I declare that this project report is my original work and it has not been presented anywhere else for the award of any degree or diploma.

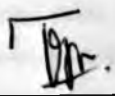
Patrick Njuguna Wanjiku

Signed: 

Date: 19/08/2010

This project report has been submitted with my permission as the University supervisor.

Elisha Opiyo

Signed:  _____

Date: 3/9/2010

ABSTRACT

In many training institutions course administration function is of significant strategic importance. Performance of the core function determines survival and success of any institution in competitive environment. The main part of course administration involves gathering, processing and distribution of information to/ from various players. Timely availability and accuracy of the same is a key success factor.

Modern life dynamics present numerous challenges which require continuous and innovative solutions. One way to address these problems is to equip workers with appropriate skills through value based flexible short courses; Training institutions' are well aware of these facts and hence their operations are distributed across various towns from where data is gathered, processed and exchanged through communication networks. Due to advancements in technology, organizations have started noticing the enormous potential benefits of applying Information and Communication Technology (ICT) to this.

Increased number of courses, tutor, learner and flexible schedule requires significant data and information flow between learning centers, learners and staff. The frequent and adhoc information exchange places heavy administrative burden on users due to information overload. The situation is compounded further by the phenomenal growth of network connectivity. Computer networks experience numerous problems ranging from congestion, connectivity and sometimes total breakdown. Agent technology hold a lot of promises if carefully deployed in some stages of information gathering and distribution.

This research project sought to identify agents' properties that can be utilized to create software agent which will serve as user's assistance at various level. Agent technology will be utilized to create a prototype Agent based service support system for course administration.

ACKNOWLEDGEMENTS

I thank God for giving me the following great people.

I would like to express my sincere appreciation to my supervisor Elisha Opiyo for his guidance throughout my research. His timely advice and direction paved the way for the accomplishment of this work .I honor him.

I wish to express my gratitude to all SCI lecturers who have laid the foundation bricks of my education. I would like to thank all the SCI technical and non teaching staff who in one way or the other helped in pursuit of this project. I wish to express my sincere gratitude to Gatheru for his support and encouragement. Onditi I salute you.

I would like to honor my mum who made all my life's achievements possible by teaching me the most important lessons in life.

Thanks to my beloved wife Winnie, son Ian, daughters Rose and Mercy for patience with me through the project. You gave me the greatest reason to succeed.

I wish to thank all my friends whom we shared a lot, just to mention P. Ndungu, E.Ndungu, Kinyanjui, Ngoru and others who always encouraged me.

Table of Contents

DECLARATION	II
ABSTRACT	III
ACKNOWLEDGEMENTS	4
CHAPTER ONE: INTRODUCTION	9
1-1 BACK GROUND OVERVIEW.....	9
1.2 PROBLEM DEFINITION.....	11
1.3 RESEARCH OBJECTIVES	11
1.4 PROJECT JUSTIFICATION	12
1.5 SCOPE AND LIMITATION	12
CHAPTER TWO: COURSE ADMINISTRATION	13
2.1 INTRODUCTION.....	13
2.2 STUDENT RECRUITMENT	13
2.3 COURSE LISTING.....	14
2.4 TUTOR RECRUITMENT.....	14
2.5 COURSE ALLOCATION AND SCHEDULING	14
2.6 EXAMINATION PROCESSING	15
CHAPTER THREE: AGENT BASE SYSTEMS.....	16
3.1 AGENTS.....	16
3.2 AGENTS CLASSIFICATION.....	17
3.3 PROPERTIES OF SOFTWARE AGENTS	19
3-5 MOBILE AGENTS SYSTEMS ENVIRONMENTS	21
3.6 MOBILE AGENT SYSTEMS SERVICES	22
3.7 DESIGN ISSUES IN MAS.....	22
3.7 SURVEY OF MOBILE AGENT SYSTEMS.....	25
CHAPTER FOUR: ANALYSIS.....	28
4.1 INTRODUCTION	28
4.2 OPERATIONS.....	28
4.3 CURRENT INFORMATION SYSTEM MODEL	32
CHAPTER FIVE: METHODOLOGY	36
5.1 INTRODUCTION.....	36
5.2 OVERVIEW OF THE SYSTEM	36
5.3 AGENT ORIENTED ANALYSIS AND DESIGN	37
5.4 SELECTED METHODOLOGY	39
5.5 THE GAIA METHODOLOGY	39
5.6 APPLYING GAIA IN ANALYSIS AND DESIGN PHASE.....	42
CHAPTER SIX: DESIGN	51
6.1 INTRODUCTION.....	51
6.2 SYSTEMS USER.....	51
6.3 SOFTWARE AGENTS	54
6.4 AB3S SOFTWARE AGENTS.....	54
CHAPTER SEVEN : IMPLEMENTATION	57
7.1 INTRODUCTION.....	57
7.2 DATA REPOSITORY	57
7.3 USER INTERFACE	57
7.5 DEPLOYMENT	60
CHAPTER EIGHT: ANALYSIS OF RESULTS.....	61

8.1	RUNNING AB3S	61
8.2	TESTING	61
8.3	REGISTERING STUDENT	61
8.4	SCHEDULING OF EXAMINATION	63
8.5	RELEASE OF EXAMINATION RESULT.	63
8.6	SUMMARY	63
CHAPTER NINE: COCLUSION AND RECOMMENDATION		66
9.1	INTRODUCTION.....	64
9.2	EVALUATIONS OF OBJECTIVES	64
9.3	CHALLENGES.....	65
9.4	LESSONS LEARNED	66
9.5	RECOMMENDATION	66
9.6	FURTHER DEVELOPMENT	66
9.7	CONCLUSION	67
10	BIBLIOGRAPHY	68
APPENDICES.....		70
APPENDICES A :	REPORTS.....	70
APPENDICES B:	AGENT CONFIGURATION INTERFACES	74
APPENDICES C	AB3S INTERACTIONS MODELS	76
APPENDICES D	AB3S SAMPLE CODES.....	79

List of figures

FIGURE 3.1	AGENTS CLASSIFICATION	18
FIGURE 3.2	COMPONENTS OF MOBILE AGENTS	22
FIGURE 4.1	INFORMATION FLOW BETWEEN CENTERS	31
FIGURE 5.1	MASE METHODOLOGY	37
FIGURE 5.2	THE GAIA METHODOLOGY	39
FIGURE 5.3	GAIA MODELS	40
FIGURE 5.4	DETAILED GAIA MODEL	41
FIGURE 5.5	FORMAT OF ROLES ATTRIBUTES.....	42
FIGURE 5.6	STUDENT ROLE.....	43
FIGURE 5.7	REGISTRATION HANDLER ROLES.....	44
FIGURE 5.8	EXAMINATION SCHEDULER ROLE	45
FIGURE 5.9	RESULT PROCESSOR ROLES	45
FIGURE 5.10	COURSE COORDINATOR ROLES	46
FIGURE 5.11	COURSE INSTRUCTOR ROLES.....	47
FIGURE 5.12	A SAMPLE INTERACTION MODEL.....	47
FIGURE 4.12	THE AGENT MODEL.....	48
FIGURE 5.13	THE SERVICES MODEL.....	49
FIGURE 5.14	THE ACQUAINTANCE MODEL	50
FIGURE 6.1	INTERRACTION IN WEB BASED SYSTEM	51
FIGURE 6.2	BASIC DATABASE TABLE DESIGN	54
FIGURE 7.1	SYSTEM CONFIGURATION	57
FIGURE 7.2	AB3S LOGIN INTERFACE.....	58
FIGURE 7.3	CENTER COORDINATOR MENU.....	58
FIGURE 7.4	PHONE TO GSM CONNECTION.....	59
FIGURE 8.1	STUDENT APPLICATION FORM	62
FIGURE 8.2	REPORT GENERATE BY AGENT	62

FIGURE 8.3	REPORT SENT TO CENTER COORDINATOR	63
FIGURE 10.1	LIST OF REGISTERED STUDENT	70
FIGURE 10.2	LIST OF STUDENT REGISTERED IN ONE OF THE CENTERS.	71
FIGURE 10.3	VISUAL DISPLAY OF AGENT IN ACTION SENDING MAILS	71
FIGURE 10.6	LIST OF TUTORS	73
FIGURE 10.8	UNIT ALLOCATION TO ALL TUTORS	74

LIST OF APPREVIATIONS

AB3S	Agent Based Service Support System
ARA	Agents for Remote action
AI	Artificial Intelligence
AUML	Agent based Unified modeling language
UML	Unified Modeling Language
CBIS	Computer Based Information System
ICT	Information and communication technology
MA	Mobile Agent
MASE	MultiAgent System Engineering
LC	Learning Center
MAS	Mobile Agent system
JVM	Java Virtual machine
SCI	School of computing and informatics

CHAPTER ONE: INTRODUCTION

1-1 Back ground overview

Recent advances in ICT technologies, such as the distributed systems, mobile computing, leads to enormous quantity of data and information processing within the Internet and other large-scale networks. The vast information presents numerous opportunities as well as challenges. The more information that becomes available electronically gives the user greater diversity and choice, but means that user must spend more time extracting and sorting the relevant information from the increasing volumes of data. Organization with affiliated business center distributed across wide geographical region depends heavily on communication channels for information and data transfer. Availability and reliability of network all time round is crucial for timely response and decision making. In case network goes down for long or is unstable, organizations' operation, image and performance is seriously affected leading to significant loses. The problems are further complicated by the fact that the final data recipients are human user. Reliance on human operator is costly, time dependent and quality result can't be guaranteed due to many human factors, significant improvements on communication devices, protocols and software have lead to improved connectivity, but optimal performance has not been realized.

There are many problems associated with large scale information collection and dissemination ranging from incorrect, incomplete and outdated to information overload. Many technologies have come in to address these problems one of which has been in applying agent technology. There has been increased research interest on applications of agent technology to application domains for which it would have a major impact. Effective Course administration is of key strategic importance for a training organization and is one of the application domains.

Course administration refers to the assembly of activities which includes among others, communication with prospective students; communication with potential student and staff involved in selection; courses listing; student and staff recruitment; course selection, units allocation and scheduling; assessment of student performance and dissemination of performance summaries. Course administration is characterized by, among others, the need for timely and accurate information; courses, student and staff dynamisms; unstructured correspondences, information overload; globalization and competitiveness. These features make course administration function suitable for the application of multi-agent solution.

A software agent is a piece of software that acts for a user or other program in a relationship of agency (http://en.wikipedia.org/wiki/Software_agent) Another working definition describe Software agent as an entity which functions continuously and autonomously in a particular environment (shahom, 1997). The concept of an agent provides a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of its user. Mobile agents refer to self-contained and identifiable computer programs that can move within the network and act on behalf of the user or another entity.

Multi-agent systems consist of a group of agents that can potentially interact with each other (Vlassis 2003) to accomplish a given task. The software agents' possess feature among others, autonomy, mobility, social ability and intelligence which making them suitable for course administration.

Timely availability of correct information enable organizations reach a diverse population and to provide good and services efficiently and probably at low cost. As organization seeks ways of increasing revenue at a minimal cost, ICT provide the much needed opportunity. For instance Distance education and e-learning offers training opportunities for students who cannot travel to main campus for their classes hence saving on administrative and travel cost. Business process outsourcing is also becoming part of business operations. Management of distributed business has been enhanced by use of ICT.

Management of organizations' data resources requires reliable communication across the network, which faces a number of problems ranging from high network traffic, congestion, connectivity, breakdown etc. To deal with increased network traffic problems, Telecommunications industry is laying down large amounts of fiber optic cables. Wider bandwidth will soon be available on the Internet backbone; this is very promising for business settings. Availability of wider bandwidth coupled with many flexible courses offered in many learning centers lead to increased volumes of data transfer which in turn introduce another problem of information overload. Accessing, sorting and processing mountains of information require many skilled personnel working through out and sometimes forget their core business function. Human users of information get overwhelmed by endless and probably useless stream of data and with time end up using trial and error information discovery methods. This approach is not only tedious and wasteful, but can be counter productive. The many network and information overload problems call for new approach to collection and processing of information. Agents technology addresses some of these problems.

Software agents make new applications possible, leading to better performance than traditional techniques under the prevailing computational and network conditions. Although it is possible to propose an alternative existing technology, Software agents have significant advantages over conventional approaches. The motivation for using software agents stems from the following anticipated benefits:

- i) Asynchronously and autonomously execution
- ii) Interaction with real time entities
- iii) Reduce network traffic
- iv) Improved human productivity and efficiency
- v) Local autonomous processing of data
- vi) Support for heterogeneous environments
- vii) Protocol abstraction

Course administration involves many complex and frequent changing rules and regulation. This is mainly because of the nature of training organizations, competition, customer and market demands which may prescribe changing guidelines to be used in selecting course, students, tutors, maintaining accurate and timely correspondence amidst numerous customer and staff demands in a very competitive business environment. Administrative personnel find it difficulty to remember and keep up- to-date with schedules or timelines and may not optimally carry out the above activities efficiently.

Therefore to handle course administrative functions, with its inherent difficulties, we aim to show the use of multi-agent technology in course administration by exploiting agents' characteristics to handle some of course administrative issues.

1.2 Problem definition

The process of course administration is repetitive, tedious, and costly and time consuming because of nature of courses, student and staff involved. The aim of this project is to explore possibilities of using software agents based applications to automate procedural and routine operations and to offload some tasks from human user. More specifically the aim of this project is to come up with an agent-based services support system that automates the course administration process, reduces the time taken to acquire, assemble, process and disperse enquiries and responses on course information to/from appropriate users and therefore lead to cost savings ,efficient course administration and improved human productivity.

Course administration process is fragile with changing course demands; diversity, locality and dynamism of students and staff, competition and globalization .If not handled properly and efficiently the effect would have a far reaching effects. The domain also requires managing enormous amount of information which is heterogeneous, distributed, unstructured due to nature of training institutions involved. The combination of these factors requires autonomous and probably adaptive technologies to implement.

1.3 Research objectives

The main aim of this project was to establish the features of software agents, operating environment to discover how the environment can be provided and features exploited to facilitate communication and improve on human productivity. This project specifically proposed to build and deploy a prototype Agent Based service support system(AB3S) that can automate the course administration process and could be applied in administration of distributed business enterprises such as private institutions, universities various departments of sparsely distributed business setting among many others.

In order to develop AB3S, the following specific objectives will be met:

- (i) To research on agent based technology and applications and utilization of software agents in the area of education and training
- (ii) Examine and evaluate existing communication technologies that support distributed organisations such as institutions, banks, retail enterprises etc.
- (iii) Examine tasks that could be offloaded from human users and assigned to software agents.
- (iv) To propose a model for implementing an AB3S for medium to large size organization.
- (v) To implement an agent based system based on the above model.
- (vi) To test and evaluate the prototype using suitable test cases

1.3 Project justification

AB3S would be used by the concerned organization to facilitate and support frequent communication both internally and externally more efficiently on time and at reduced costs. To facilitate communication AB3S would use available communication tools that is email and mobile phone ,which are readily available .AB3S would support course administration process and free course administration personnel so as to engage in more value adding activities.

AB3S will be responsible for most of courses administrative activities hence it would reduces administrative cost, user response time and reduces number of queries.AB3S would demonstrate to practitioner in agent research the possibility of building agent based applications for real-world applications.

1.4 Scope and limitation

The scope of this project include examination of agents features, requirement analysis, design and implementation of an agent based solution for course administration. Course administration functions covered include student and staff recruitment; course and units listing and registration; submission of draft examination and scheduling of the same; Submission of raw marks by tutors and release of examination results.

The prototype is limited to course administrative functions listed above and doesn't support online registration or other aspects of courses such as evaluation and user feedbacks; learning process among many others.

CHAPTER TWO: COURSE ADMINISTRATION

2.1 Introduction

Course administration refers to a set of activities carried in training institution. Training involves not only identifying and addressing pedagogical challenges but management challenges as well. These includes, but not limited to establishment and management of curriculum; working with trainers, external institution and administrator, handling both internal and external communications to and from students and training governing bodies.

This project focus on specific aspects of course administration which includes; course, unit listing and advertisement ;student and staff recruitment; course unit scheduling and allocation; examination setting, administration and control as well as posting result to student and administrator.

Rapid developments in information and communication technologies (ICT) in recent years have resulted in significant changes in the way the world operates and communicates. This in turn has had an impact on educational and training needs, both in terms of the content and the delivery of educational and training services.

ICT has presented numerous opportunity among them distance and e-learning; affiliated and distributed training institution; effective and efficient mode of communication. There are number of challenges facing use of ICT, but benefit out weights limitations. As ICT evolves new opportunities as well as challenges emerges and requires continuous evaluation to take advantage of ICT and addresses challenges.

2.2 Student Recruitment

Private training organizations always get student they need using traditional recruitment procedure which is characterized by three phase: Course advertisement , applicant selection and admission. Courses advertisement involves use of communication media such as Radio, TV, newspaper, email, text messages among many other to inform prospective students of available and new courses. Course advertisement is a critical task for every institution. In effort to recruit potential student other methods may be applied which includes but not limited to:

- (i) Marketing, Advertising, Publication, Promotional Gifts and Material
- (ii) Coordination with External Partners
- (iii) Coordination with Internal Partners (Students, Alumni, Faculty and Staff)
- (iv) Events Management (Fairs, Conferences, Open Days)
- (v) Global Strategy of Recruitment Activities
- (vi) Evaluation and Market Research (e.g., Student Interviews)

Successful advertisement efforts lead to higher enrollment in various programs. This helps the education institutions to sustain their resources and maintain high level of educational services throughout its wide variety of academic programs.

Course advertisement encounters difficulty challenges ranging from delayed delivery of content; diversity of target group; poor or delayed or no response; constrained budget; lack of trained marketing personnel; competition among many others.

The selection phase involves identification of successful applicant per center and course against a selection criterion. Successful applicants are informed through available and appropriate media. Selection phase encounter numerous challenges which include; volumes of valid and invalid applications; flexible but strict selection criterion; trial and error applications; delayed or unstructured response; systems failure; staff issues ; delayed response to successful applicant; unmatched response to admission numbers; heavy dependence on service provider, cost among many others.

The admission phase involves registration of successful applicant, attaching them to faculties of choice and enrolling them for various course units offered at various centers. This phase admit number of challenges which includes: delayed admission; diversity of units, prerequisite ; attrition to other course; student flexible schedule; lack of or inadequate information; constrained resources; lack of quorums or trainers in certain areas; Poor student support; Strict schedule among many others.

To address these challenges decisive methods of selection and communication need to be established.

2.3 Course listing

The Course Listing allows student, tutor to search for and get details about available courses for each semester. Courses or new units of courses may be added, discontinued or changed after the Course Listing for a particular semester. Additional information about new courses is available from the departments offering them. Proper and accurate course listing plays an important role in supporting student, tutor as well as administrator.

To administrator course listing provide guideline to student on available options. Upon selection and registering for units the information aids decision making on matters of allocation, scheduling and staffing. A number of challenges are encountered ranging from outdated courses, lack of interest; staffing problems; routine selection of units among others.

2.4 Tutor recruitment

Attracting, Selecting and maintaining right tutor s is critical to success of any institution in terms of content delivery, institutional image, student support and overall course administration. Every institution faces many challenges in these area rights from selection, retention and management. These problems are compounded further by staff dynamisms, locality as well as inherent human factors.

2.5 Course Allocation and Scheduling

Course allocation function involves identification of course unit on offer, number of student enrolled per center and available tutor after which unit are scheduled for leaning and evaluation. This function is complicated or tricky

because of number of factors which includes: units on offer; failure rate; progression principle; course content duplication; cross link between departments; market demand as well as fall out after registration.

2.6 Examination Processing

Examination processing functions include: identification of unit-tutor allocation and schedule; submission of draft examination; recording of students' score per subject; consolidation of centers' semester units score; consolidation of all centers' semester units score, generating pass list; posting of results summaries to coordinator and student submission; scheduling semester, special and retake examination.; responding to all tutor and student enquiries.

There are numerous challenges encountered in examination processing among them: diversity of tutor and course; strict and overlapping schedules; poor or delayed correspondence and audit of the same; staff overloads; poor or unreliable communication and poor record keeping.

CHAPTER THREE: AGENT BASE SYSTEMS

3.1 Agents

There exist various definitions of software agents, but there is no universally accepted one, some researcher define agent based on properties they exhibit ,others on execution environment while others base their definition on agent specific action or tasks.

According to Genesereth (1994) software agents are entities that can communicate in an agent communication language while Meas(1995) define software agents as computer systems in a complex environment that realize a set of tasks and goals they were designed for. Other define agents as Computer programs that act autonomously on behalf of a person or organization (OMG MASIF)(OMG, 2001) while FIPA(2001) define it as Computational processes implement an application's autonomous communicating functionality

According to (Wooldridge, 2002) an agent is as a computer system that is capable of independent action on behalf of its user or owner. It states further that an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

(Wooldridge and Jennings 1995) also define an agent as a self-contained problem-solving entity (implemented in hardware, software or a mixture of the two), which exhibits the following properties: autonomy, social ability, responsiveness and proactiveness.

According to (Grabner et al. 2000), a software agent is a software program that can do some defined action for a user, which has some kind of intelligence. To give the ability to do part of its functionality autonomously and which can interact with its environment. This definition is based on three properties of agency: intelligence, interaction and autonomy. (Weiss 2001) defines an agent as an autonomous computational entity (autonomy), which interacts with its environment (reactivity) and other agents (social ability) in order to achieve its own goals (proactiveness).

The response of some agent researchers to this lack of definition has been to invent yet some more synonyms. So we now have synonyms including knowbots (i.e. knowledge-based robots), softbots (software robot), taskbots (task-based robots), userbots, robots, personal agents, autonomous agents and personal assistants there are some good reasons for having such synonyms. Firstly, agents come in many physical semblance: for example, those that inhabit the physical world are called robots; those that inhabit vast computer networks are sometimes referred to as softbots; those that perform specific tasks are sometimes called taskbots; and autonomous agents refer robots which operate in dynamic and uncertain environments. Secondly, agents can play many roles, hence personal assistants or knowbots, which have expert knowledge in some specific domain. Furthermore, due to the multiplicity of roles that agents can play, there is now an excess of adjectives which precede the word agent, as in the following drawn only from Kingis (1995) paper: search agents, report agents, presentation agents, navigation agents, role-playing agents,

management agents, search and retrieval agents, domain-specific agents, development agents, analysis and design agents, testing agents, packaging agents and help agents

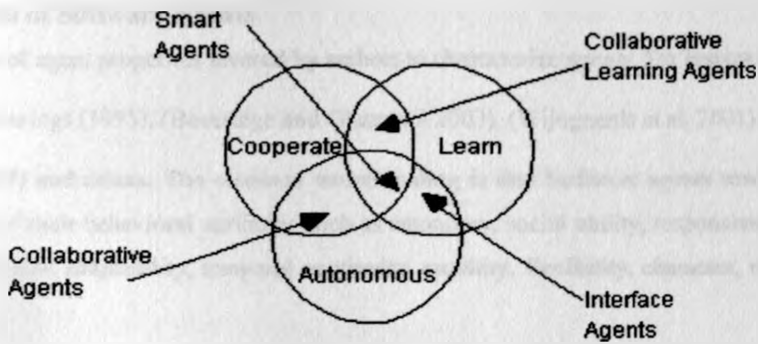
3.2 Agents Classification

There are many attempts to classify existing software agents according to various dimensions. The attempts has been made in order to accommodate all the agents.

Firstly, agents may be classified by their mobility, i.e. by their ability to move around some network. This yields the classes of *static* or *mobile* agents.

Secondly, they may be classed as either *deliberative* or *reactive*. Deliberative agents derive from the deliberative thinking paradigm: the agents possess an internal symbolic, reasoning model and they engage in planning and negotiation in order to achieve coordination with other agents. Works on reactive agents originate from research carried out by Brooks (1986) and Agre & Chapman (1987). These agents on the contrary do not have any internal, symbolic models of their environment, and they act using a stimulus/response type of behavior by responding to the present state of the environment in which they are embedded (Ferber, 1994). Indeed, Brooks has argued that intelligent behavior can be realized without the sort of explicit, symbolic representations of traditional AI (Brooks, 1991b).

Thirdly, agents may be classified along several ideal and primary attributes which agents *should* exhibit these: autonomy, learning and cooperation. *Autonomy* refers to the principle that agents can operate on their own without the need for human guidance, even though this would sometimes be invaluable. Hence agents have individual internal states and goals, and they act in such a manner as to meet its goals on behalf of its user. A key element of their autonomy is their pro activeness, i.e. their ability to take the initiative rather than acting simply in response to their environment (Wooldridge & Jennings, 1995a). *Cooperation* with other agents is paramount: In order to cooperate, agents need to possess a social ability, i.e. the ability to interact with other agents and possibly humans via some communication language (Wooldridge & Jennings, 1995a). It is possible for agents to coordinate their actions without cooperation (Nwana *et al.*, 1996). Lastly, for agent systems to be truly smart they would have to *learn* as they react and/or interact with their external environment. The learning may also take the form of increased performance over time. According to Nwana(1996) four types of agents are derived from agents attributes these are: *collaborative agents*, *collaborative learning agents*, *interface agents* and *truly smart agents*. The distinctions are *not* definitive



1 Figure 3.1 Agents Classification[Hyacinth S. Nwana,1996]

Fourthly, agents may sometimes be classified by their roles (preferably, if the roles are major ones), e.g. world wide web (WWW) information agents. This category of agents usually exploits internet search engines such as WebCrawlers, Lycos and Spiders. Essentially, they help manage the vast amount of information in wide area networks like the internet. Information agents may be static, mobile or deliberative.

Fifthly, we have also included the category of *hybrid* agents which combine two or more agent philosophies in a single agent.

There are other attributes of agents which we consider *secondary* to those already mentioned such as versatile; veracity; temporally continuous; emotional attitudes ; *mentalist* attitudes or notions such as beliefs, desires and intentions

These definitions are slightly different from one another. There is however no complete standard/consensus definition of an agent ,but they reflect important types of agents, such as autonomous, adaptive, reactive, mobile, cooperative, interactive, and delegated.

In this project we take software agent According to(krupansky,2004) a software agent is a computer program which works toward goals in a dynamic environment on behalf of another entity (human or computational), possibly over an extended period of time, without continuous direct supervision or control, and exhibits a significant degree of flexibility and even creativity in how it seeks to transform goals into action tasks

3.3 Properties of Software Agents

There are a number of agent properties favored by authors to characterize agents. For instance,

(Wooldridge and Jennings (1995), (Beveridge and Glasspool 2003), (Wijngaards et al. 2001) and

(Grabner et al. 2000) and others. The common understanding is that Software agents tend to be characterised in terms of a number of their behavioral attributes such as autonomy, social ability, responsiveness and proactiveness. Other properties include adaptability, temporal continuity, mobility, flexibility, character, veracity, rationality, and intelligence.

(a) Autonomy

This means that agents should be able to perform the majority of their problem-solving tasks without direct intervention of humans or other agents, and they should have a degree of control over their own action and their internal state, (Castelfranchi 1995), (Wijngaards et al. 2001) and (Grabner et al. 2000).

Once launched an agent should be able to operate independently from their user, that is, autonomously in the background. An agent needs to have control over its actions so that it can determine what to do when an action succeeds or fails.

(b) Social ability

To effect changes or interrogate their environment, an agent must possess the ability to communicate with the outside world. This interaction can exist at a number of levels depending upon the dispatch of the agent, but typically an agent would need to communicate with other agents, the local environment and users.

(c) Reactivity

Agents need to be able to perceive their environment and respond to changes to it in a timely way. Agents need not only to be aware of their environment, but they need to be aware of what the state and changes in that environment mean and how to react to them.

(d) Pro-activity

Agents need to be able to exhibit pro-activeness, that is, the ability to cause actions in order to achieve their goals by taking the initiative. This means that an agent needs to appreciate the state of their environment and to decide how best to fulfill their mission target.

(e) Intelligence

This means that agents are capable of learning, have knowledge, can perform complex tasks and can reason about and with this knowledge, (Wijngaards et al. 2001). The agent has some specialized knowledge in one or more application fields,

(f) Mobility

This is the ability of an agent to change its location. According to (Kurt Rothermal,1998) mobility is motivated by the rise and rapid growth of a networked computing environment and the need for techniques to exploit this huge resource.

Agents do not necessarily need to possess all the properties but can exhibit a blend of some or all of the above properties.

3.4 POTENTIAL BENEFITS OF SOFTWARE AGENTS

Various authors (Nwana, 1996, Sycara 1998, Whitney 1999) identifies various potential benefits that could be derived from exploitation of software. Implementing an agent based system would result in realization of some or all of the following benefits.

(a) Scalability

The number and capabilities of agents working on a problem can be altered (Sycara, 1998). These allow new agents to be added to the system to enhance system functionality as long as they abide by rules spelt out in the system.

(b) Robustness

The system is able to tolerate uncertainty, because suitable information is cooperatively exchanged among agents, (Sycara 1998). When agents leave the system or die other agents that they were cooperating with are notified enabling them to engage any contingency measures

(c) Maintainability

System composed of multiple components (agents) is easier to maintain because of its modularity, (Sycara 1998) (Nwana 1996). The agent abstraction can be used to break problems into societies of cooperating autonomous software entities each carrying out a well-defined functionality. Therefore any changes made to the system are mostly made on the component agents, not the entire system.

(d) Flexibility

Flexibility-this is because agents with different abilities can adaptively organize to solve the current problem, (Sycara 1998). Agents can use different artificial intelligence techniques, such as learning, to improve their problem solving ability and cope with environmental changes, such as the sudden disappearance of counterpart agents from the agent society.

(e) Conserving bandwidth and reducing latencies

By migrating to the location of a needed resource an agent can interact with the resource without transmitting intermediate data across the network, conserving bandwidth and reducing latencies. By migrating to the location of a user, an agent can respond to user action rapidly. The agent can continue its interaction with the resources or user even if network connections go down temporarily. These features make mobile agents attractive in mobile computing application.

(f) Fault tolerance

According to (Aderounmu and Oyatokun 2006) Fault tolerance is the ability of a system to respond gracefully to an unexpected hardware or software failure. A fault tolerant system, as in computer networks, has the ability to continue operation in the event of a network failure. Modular nature of agents enable handling of anomalies locally and not to propagate them to the whole system

(g) Work offloading

Software agents allow traditional clients and servers to offload work to each other and to change who offloads to whom according to the capabilities and current loads of the client, server and network.

(h) Disconnected operations

Software agent can operate asynchronously and autonomously from the process that created them, after being dispatched. Mobile devices, which need continuous access of fixed network, often suffer from fragile and low bandwidth connects. In such cases they can embed their task in MAs, dispatch them, and then reconnect later to collect these agents.

(i) Reuse

Agents' functionality in specific agents can be reused in different agent teams to solve different problems

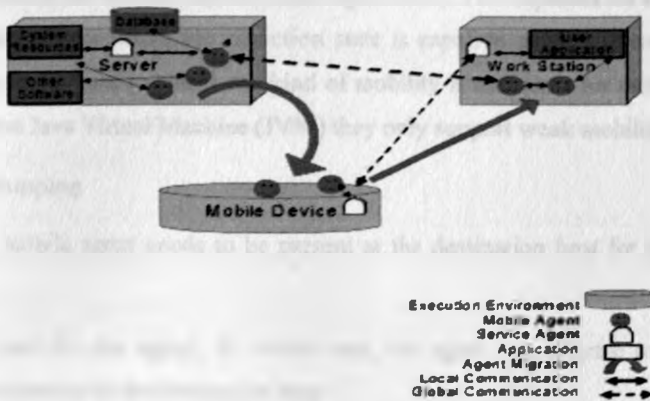
(j) Dynamic protocol

The rapid growth of the internet has also increased the number of protocols and data formats for data exchange between computers. A computer generally supports only a limited number of protocols. If a particular protocol is missing in order to access, view or process some received data, the protocol must be installed manually. A mobile agent permits dynamic protocols, i.e. new protocols to be installed automatically and only as needed for a particular interaction.

Although technologies such as process migration and object mobility have been present for some time, they have not become widespread for a number of reasons:

3-5 Mobile Agents Systems Environments

To realize a mobile agent in practice, support from the underlying distributed network of machines is required. A developer should be able to write a MA using some programming language, a MA once created needs an **execution environment to run**, it needs to send and receive messages from its surroundings, it needs resources for storing data, it may need to persistently store itself or migrate to some other node. Its actions need to be controlled, it has a definite life-cycle, it may need to work in collaboration with other MAs and it might be of significance to trace its movements and in some cases it may require protection. The Framework that provides mechanisms for supporting these facilities for mobile agents is called *mobile agent framework*. The figure below represent different components of a mobile agent's framework.



2 Figure 3.2 Components of Mobile Agents

3.6 Mobile Agent Systems Services

Mobile agent's framework provides the following services for Mobile Agent;

- **Life Cycle** These are Services to create, destroy, start, suspend, stop etc.
- **Navigation** This service is responsible for transporting an agent (with or without state) between two computational entities residing in different locations.
- **Communication** This is between agents and between agents and other entities. The naming and the addressing mechanisms followed in the system.
- **Security** Ways in which agents can access network resources, as well as ways of accessing the internals of the agents from the network

3.7 Design Issues in MAS

Mobile agents require services from mobile agent systems .A Mobile Agent Framework is an infrastructure (Systems) that implements the agent paradigm. The various design issues in designing such frameworks are:

(i) Mobility Model

The fundamental requirement in a Mobile Agent System is its ability to transfer an MA from one host to another. Whenever the migration occurs, the agent is deactivated, its state is captured, and this state is transferred to the new site along with the agent code. On the new site the state is again restored and the agent is reactivated. Depending upon the nature of state transmitted, mobility can be of two types:

(ii) Strong Mobility:

If both the data and the execution state of the agent are transmitted, it is the case of strong mobility. The destination server can restart the execution of agent precisely from the point where the execution was stopped on the originating host.

(iii) Weak mobility:

In this case the state of the agent is captured at a higher level. This captures the execution state only at function-level in contrast the earlier case where the execution state is captured at instruction level. Since the mobile-agents are under the direct programmer's control, the kind of mobility is sufficient for most of the applications. Since most of the frameworks use Java Virtual Machine (JVM) they only support weak mobility.

(iv) Code Shipping

The code of the mobile agent needs to be present at the destination host for its successful restart. This code can either be:

- (a) **Carried by the agent**, in which case, the agent can migrate to any host providing the execution environment at the destination host
- (b) **Pre-installed on the destination host** This is better for security reasons as no foreign code is allowed, but this restricts the use of MAs only to pre-defined set of machines
- (c) **Available on a code-base server**, from where it can be downloaded on-demand.

(v) Agent Naming and Addressing Mechanisms

Agents need to be named and located to enable inter-agent communications or remote agent management. Naming of the agent can be location dependent or independent. Different agents running parallel on different hosts may need to exchange temporary results and synchronize. One of the convenient ways of doing it is for agents on different hosts to use a common shared naming server. The other alternative is to locate the agent from the host from which it originated and which is also keeping the logging information about the current location of such agent.

There are various schemes for locating mobiles agent and delivery of messages between them which includes:

Brute Force: Agent is located by searching it in multiple destinations. Searching can be parallel or sequential.

Logging: An agent is located by following its trail information, indicating its next destination, left in every agent server it already visited. Trail information for the disposed agents can be garbage-collected according to, for example, the expired time or explicit notification by agents.

Registration : An agent updates its location in a predefined directory server that allows agent to be registered, unregistered or located. Other agents use the directory to locate the agent. In practice, communicating agents need to agree in advance upon a naming server. Such agreement can be simplified by adopting an architecture in which every agent server is associated with one available naming server.

(vi) Agent tracking and message delivery

Agents sometimes need to be tracked for sending them messages or controlling them remotely. There are two basic methods:

Locate-and-Transfer: An agent is located after which the message is transferred directly to it; in this case two separate phases are used.

Forwarding : Locating a receiver agent and delivery of message to it are both done in a single phase Forwarding is more appropriate than locate and transfer methods especially for small message.

(vii) Agents' communication

They need to interact with Execution Environments (EEs), resources/ objects, other agents or users to achieve their goals.

(viii) MA Security issues.

Security is an important consideration in an open network like the Internet. It is important to safeguard both the execution environment as well as the mobile agents from any undesirable effects. The different security issues relevant to mobile agents are described in the table below.

Attacker	Attacked	Attack
Arriving agents	Host	Access and corrupt the host's local files, resources.
External third party	Host	Send a huge number of agents to the host to tie up all resources or even crash system
Agents	Agents	Access private information or to crash agents to stop fulfilling its task
Third party	Agent	Alter exchanged messages for its own benefit

Table 3.1 MA security consideration

Mobile agent systems have to provide different kind of security mechanisms to detect and guard against these attacks. These are privacy, integrity and authentication mechanisms and authorization mechanisms.

3.7 Survey of Mobile Agent Systems

(a) Aglet

Aglet models the MA to closely follow the applet model of Java. It is a simple framework where programmer overrides predefined methods to add desired functionality. An Aglet is defined as a mobile Java object that visits Aglet-enabled hosts in a computer network. Aglet runs in its own thread of execution after arriving at the host, so it is attributed as *autonomous*. It is also *reactive* because it responds to incoming *messages*.

The complete Aglet object model includes additional abstractions such as *context*, *proxy*, *message*, *itinerary*, and *identifier*. These additional abstractions provide Aglet the environment in which it can carry out its tasks. Aglet uses a simple proxy object to relay messages and has a message class to encapsulate message exchange between agents. However, group-oriented communication is not available, and the choice of using a proxy to relay a message may not be a scalable solution in a high-frequency transport situation. By modeling the MA as a Java object, the designers of Aglet leverage the existing Java infrastructure to take care of platform dependent issues and to use existing mobile code facility of Java.

(b) Agent Tcl

The designers of Agent Tcl do not formally specify an MA model. Instead, MA is understood as a program that can be written in any language and that accesses features that support mobility via a common service package implemented as a server. This server provides MA-specific services such as state capture, transfer facility, and group communication, as well as more traditional services such as disk access, screen access, and CPU cycle. The philosophy was that all functionalities an agent ever wants are available in the server. Agent mobility then only concerns closure, which is the Tcl script. There are no additional codes to load. In Agent Tcl, the state capture of an agent is handled automatically and transparently to the programmer.

(c) Agents for Remote Action (ARA)

The Agent for Remote Action (ARA) system is similar in concept to Agent Tcl in that it has a core service layer supporting multiple languages through interpreters. ARA aims for seamless incorporation of "mobile programming" to the existing world of programming practice. In ARA, "the mobile agent is a program that is able to move at its own choice and without interfering with its execution, utilizing various established programming languages." The ARA agent moves between and stays at *places* where it uses services provided by the host or other agents. ARA agents are considered normal programs in all other aspects, working with file system, user interface, network interface, and other well-known computing facilities. In addition, ARA also formulates the concept of server agents, the services with which the MA would interact, as static compiled objects located at network nodes. ARA agents run within an interpreter, interfacing with a core language-independent set of services. The core services include resource management, mobility, and security among many others. The language-dependent features, such as how state capture is handled and correctness checking, are the responsibility of the interpreter, whereas providing access to the underlying Operating System services and other MA-specific services is the responsibility of the core. ARA

agents are executed in parallel threads, while some of the internal ARA core functions can be executed as separate processes for performance reasons.

(d) **Concordia**

Concordia is another MA framework built on Java. In Concordia an agent is regarded as a collection of Java objects. A Concordia agent is modeled as a Java program that uses services provided by a collection of server components which would take care of mobility, persistence, security, communication, administration, and resources. These server components would communicate among themselves and can run in one or several Java virtual machines; the collection of these components forms the AEE at a given network node. Once arriving at a node, the Concordia agent accesses regular services available to all Java-based programs such as database access, file system, and graphics, as in Aglet. Like ARA, Concordia specifies a service bridge to provide access to legacy services. A Concordia agent is considered to have internal states as well as external task states.² The internal states are values of the objects' variables, and the external task states are the states of an itinerary object that would be kept external to the agent's code. This itinerary object encapsulates the destination addresses of each Concordia agent and the method that each would have to execute when arriving there. The designers of Concordia claim that this approach allows greater flexibility by offering multiple points of entry to agent execution, as compared to always executing an "after-move" method as in Agent Tcl, Aglet, or ARA. This concept of an externally located itinerary is similarly supported in Odyssey via Odyssey's task object. However, the infrastructure for management of these itinerary objects was not clear from the publicly available literature on Concordia.

(e) **Mole**

In Mole, the agent is modeled as a cluster of Java objects, a closure without external references except with the host system. The agent is thus a transitive closure over all the objects to which the main agent object contains a reference. This island concept allows simple transfer of agent without worrying about dangling references. Each Mole agent has a unique name provided by the agent system, which is used to identify the agent. Also, a Mole agent can only communicate with other agents via defined communication mechanisms, which offer the ability to use different agent programming languages to convert the information transparently when needed.

A Mole agent can only exist in a host environment call *location* that serves as the intermediate layer between the agent and the OS. Mole also supports the concept of *abstract location* to represent the collection of distributed physical machines. One machine can contain several locations, and locations may be moved among machines. Mole limits the abstract location to denote a configuration that would minimize cost due to communication. As in ARA, Mole directly proposed the concept of a system agent which has full access to the host facilities. It is through interacting with these system agents that a given Mole agent (mobile) achieves tasks. A Mole MA can only communicate with other agents (systems and MAs) and has no direct access to resources.

(f) **Voyager**

Most MAS have taken advantage of Java language features to implement MA systems (Aglet, Concordia, Mole, and Odyssey) However, none has yet represented a state of tight integration with Java like Voyager. Proclaimed as an

agent-enhanced object request broker (ORB) in Java, Voyager offers several advanced mechanisms that could be used to implement MA systems. The Voyager agent model is also based on the concept of a collection of Java objects; it does have an *Agent* class that developers can subclass to implement Voyager-style MA. However, the product goes one step beyond to provide arbitrary remote object construction, a facility for moving objects (not just agents), and a host of other communication and infrastructure services that can be used to implement arbitrary MA systems.

The Voyager agent is designed to take advantage of the Voyager ORB features which make extensive use of Java's reflection mechanism. A Voyager agent can communicate by calling methods or using Voyager Space technology, a group event and message multicast facility that Object Space claims to be more scalable than simple communication mechanisms in Aglet or Concordia. Voyager supports a more extensive set of control mechanisms, such as more flexible instructions on how the agent should terminate itself than the fixed or explicit instructions on termination as in Aglet, Concordia, or Odyssey. As in Agent Tcl or ARA, inter-agent and high-level communication is reserved as application-level features indirectly supported with more primitive mechanisms provided. The Voyager agent model is essentially the same in concept as in Mole, Aglet, Concordia, and Odyssey systems, which is a collection of Java objects. The designers of Voyager also leave the decision on the complexity of an agent to users. There are no metrics to suggest how the complexity of a mobile program would break the system; instead, the user is led to believe that arbitrary complexity of any degree can be achieved.

CHAPTER FOUR: ANALYSIS

4.1 Introduction

Many private training institutions have students in a number of teaching and Learning Centers (LC) which are sparsely located in a number of towns across the country. All LCs' (including main one) activities are managed by registrar. Each LC is autonomous in terms of financial management, but all remit accreditation fee based on their memorandum of understanding. LC offer specialized courses based on their customer base although some are common to a number of LC.

There exists stiff competition with other training institutions for the same customers despite the fact that some institutions are government funded. For an institution to survive and guarantee return on investment it has to be innovative in terms of courses offered, promotion and administration of the same.

New technological innovation presents opportunities as well as challenges; there is a need to exploit new opportunities by launching innovative and flexible products. New and existing technologies should be harnessed to overcome challenges faced. Working solutions can be derived by getting a blend of technologies.

Distance learning has gained root in modern society due to dynamics of life; rapid development in information communication and technology and appreciation of internet as media of communication have changed the way of life and interaction; training institutions have not been left behind. New learning demands have been created by exploding information, accelerated competition and technology essential which is shortening learning time and shrinking space between learners and educators hence training functions need not be localized. Educators in this globalized economy are faced by many challenges and are expected to be flexible, innovative and supportive to meet fast-changing consumer and organizational needs.

4.2 Operations

Students apply for courses offered at a LC either through centers' coordinators or to Registrar. All applications are processed and successful applicants are informed through post or from various information desks. Successful applicants enroll for classes at center of their choice where they study, are evaluated and receive performance results. A registered student enrolls for units on offer depending on course requirements and individual needs. Units' registration information is needed for administration of the unit such as allocation to tutors and scheduling of the same. Timely availability of such information is very crucial. A registered student may transfer from one center to another and transfer credit as they relocate to new center.

Registrar decides on qualifications of tutors for all LCs this is to guarantee quality. Registrar receives a list of applicants and creates a pool of tutors from which the center coordinator can use to allocate course units. Learning center coordinators allocate units to tutors and inform them of the proceedings so as to prepare for the same. The staff prepares learning and evaluation materials for all courses offered in all LCs.

There are numbers of tutors, full-time and part-timers in each LC depending on courses offered and number of students. The role of a tutor in LC is summarized below.

- Teaches courses assigned and registered for in the LC

- Set drafts examinations for unit allocated and submit the same to registrar.
- Administer examinations. mark scripts, record and submit raw marks to centers' course coordinator for unit(s) taught

Each LC has a course coordinator responsible for the following

- Assist student in registration process by recording all application details
- Receive course information from registrar.
- Post course schedules and details to student and tutor.
- Conduct course needs assessment for unit resources allocation.
- Schedule classes.
- Monitor student progress.
- Post unit progress to registrar.
- Allocate unit to tutor.
- Evaluate tutor performance.
- Receive examination schedule, organize invigilation and post examination time table to student and tutor.
- Monitor tutor progress in marking assignment and examination.
- Receive marks from individual tutor and consolidate them.
- Post LC marks to registrar.
- Receive examination results from registrar and post to student.
- Address all tutor and student issues relating to course.
- Handle all student queries in liaison with registrar.

LC coordinator is the link person with the main campus and the students as well as prospective students. The enormous role of coordinator requires frequent and constant exchange of information between learning centers and /or main campus.

Institution registrar is central in management of all courses both in the main campus and in the LC. The following list summarizes responsibilities of registrar.

- Post announcements about upcoming courses and those offered in all LC.
- Post details of cause on offer
- Post details of courses schedules.
- Register new and continuing student every semester.

- Liaises with LC to established training needs for each LC.
- Monitor student per center enrollment.
- Receive student per unit enrollment from each LC for examination preparation.
- Schedule unit examinations for each LC
- Monitor examination setting progress.
- Schedule all examination and post to LC
- Submit examination registered for to LC
- Receive examination results from all LC
- Consolidate results from all LC.
- Analyze student performance from all learning center per course, moderate results, process pass list and post results to LC.
- Administer courses progress in all LC
- Addresses all LC and student issues.

Prospective student may apply for course offered directly to registrar or through LC coordinator .Upon admission student enroll for units of their choice depending on their needs and course requirements. Student sit for examination for units enrolled for, check on their performance before proceeding with. On completion of a course a student may apply for another course.. Figure 5.1 below represents the current information flow between LCs and main

campus.

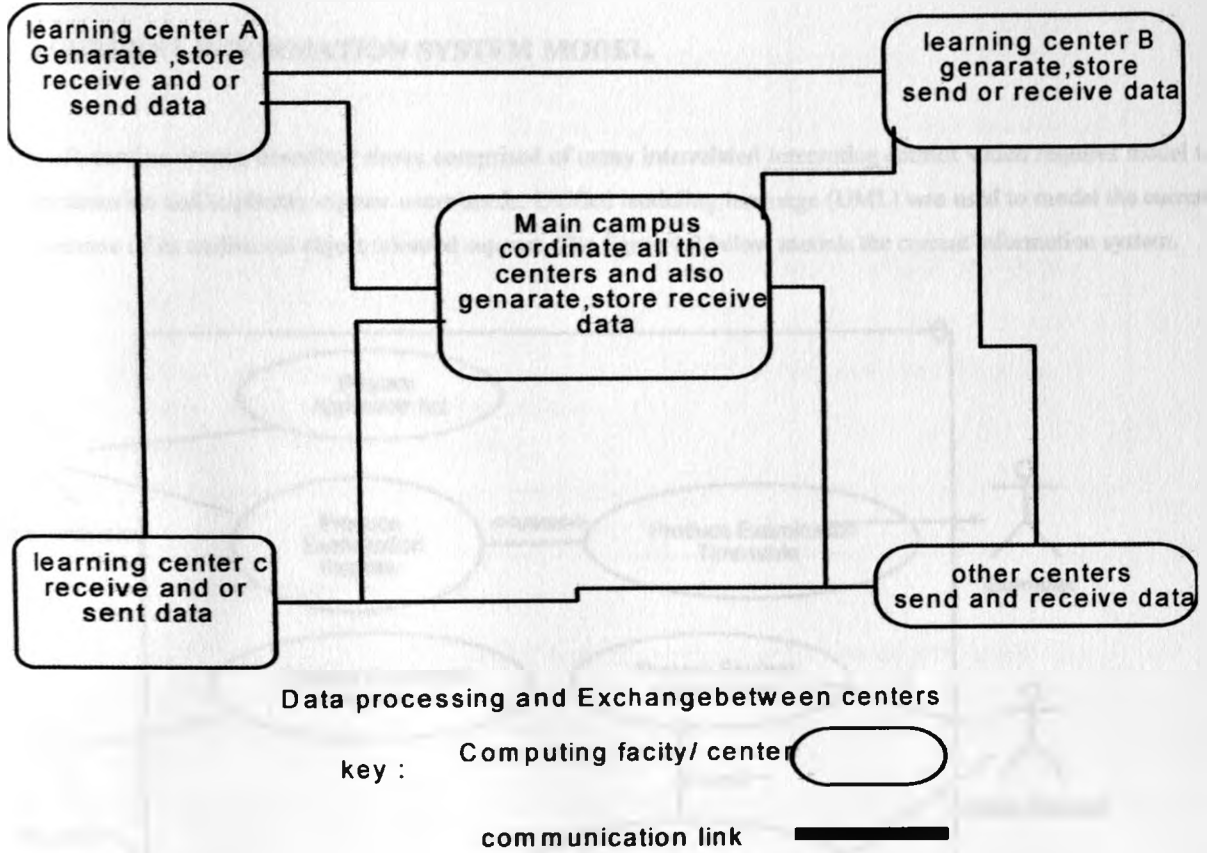


Figure 4.1 Information flow Between Centers

Currently IU doesn't have an information system to facilitate course administration. To address data storage and processing a Computer Based Information System (CBIS) is suggested. The CBIS is to record data and communications to and from all LC, students, academic staff and registrar. The proposed CBIS would enhance information storage and retrieval from storage media.

Unfortunately the role of LC coordinators and registrar is enormous and would require initiating all information retrieval explicitly and monitoring all responses from the CBIS. The problem is compounded further by unreliable communication channel and financial autonomy of centers. This direct-manipulation method of interaction is laborious (leading to information overload), time consuming and costly means of information filtering and retrieval.

4.3 CURRENT INFORMATION SYSTEM MODEL.

Current information system described above comprised of many interrelated interacting entities which requires model to aid comprehension and explicitly capture users needs. Unified modeling language (UML) was used to model the current system because of its traditional object oriented support. The figure 4.2 below models the current information system.

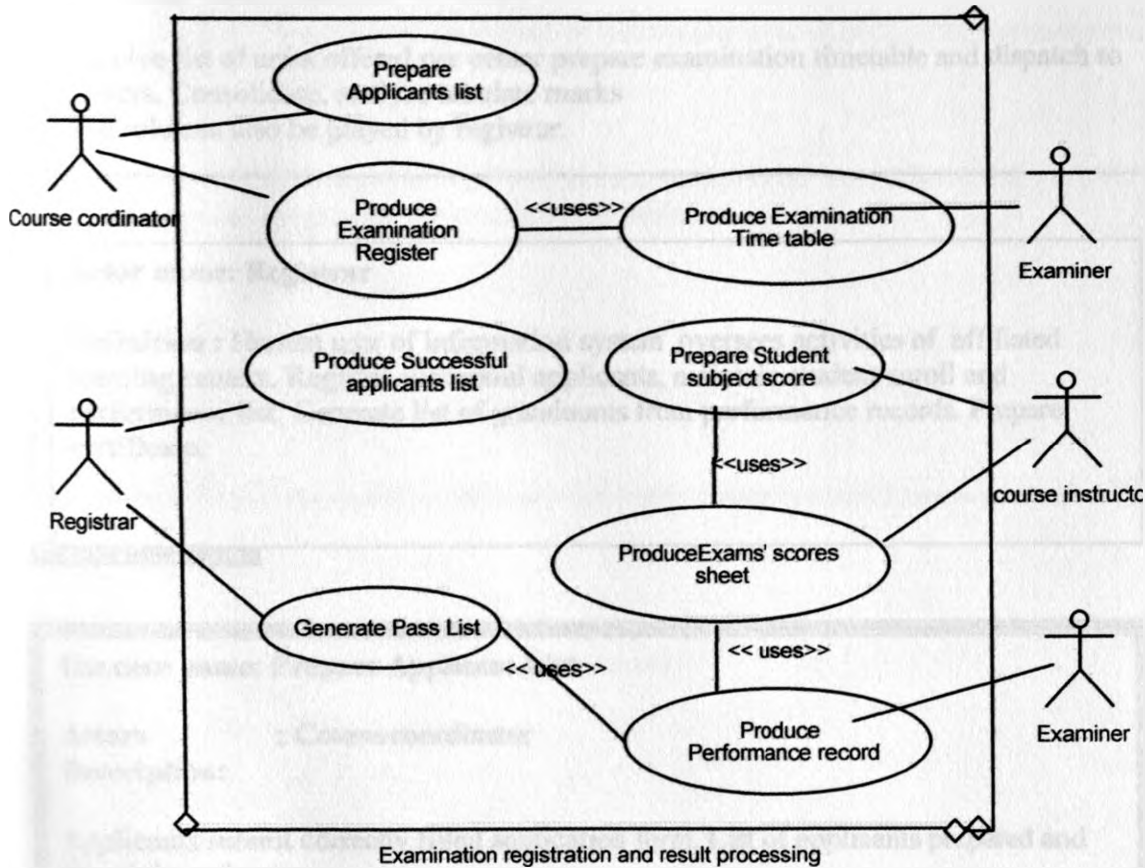


Figure 4.2 Information flow in current system

ACTOR DESCRIPTION

Actor name: Course coordinator

Definition : Human user of information system in charge of learning center. Receives applications, prepare list of units enrolled, Gathers student marks and submit to examiner.

Receive and communicate information on examinations e.g Exams offered, time table center rating etc.

Actor name: Course Instructor

Definition : Human user of information system in contact with student. Administers examinations, record unit scores.

Actor name: Examiner

Definition : Human user of information system stationed in main campus in charge of Learning centers examination matters.

Receive list of units offered per center prepare examination timetable and dispatch to centers. Consolidate, analyze tabulate marks
The role can also be played by registrar.

Actor name: Registrar

Definition : Human user of information system ,oversees activities of affiliated learning centers. Register successful applicants, maintain student enroll and performance list. Generate list of grandaunts from performance records. Prepare certificate.

Use case descriptions

Use case name: Prepare Applicant List

Actors : Course coordinator

Description:

Applicants submit correctly filled application form. List of applicants prepared and stored for reference.

Use case name: Produce Successful applications List

Actors : Registrar, Course coordinator

Description:

Applicants details processed and list of successful applicant sorted per center. List submitted to appropriate.

Use case name: Produce Examination register

Actors : Course coordinator, examiner

Description:

Coordinator prepare examinable units list and submit to examiner who prepare units' Examination for centers

Use case name: Produce Examination timetable

Actors : Examiner

Description:

Examination schedule prepared and submitted to coordinator for posting.

Use case name: Produce Student subject score

Actors : Course instructor

Description:

Course instructor enters students' score for each unit in a common mark sheet.
Preliminary examination scores' sheet prepared and submitted to examiner.

Use case name: Produce Student performance record

Actors : Examiner

Description:

Examiner consolidates, tabulate and analyze result and generate performance record and submit to registrar. Sort out performance record per center and dispatch

Use case name: Prepare Pass List

Actors : Registrar

Description:

Receive Performance list and prepare pass list, grandaunt list, progress report and certificates for posting.

There exist a number of user functional and non functional requirements which could not be captured in the model this includes

- (i) Large volume of data transfer between computing nodes, the volume is expected to grow significantly with introduction of customized short courses.
- (ii) Ad hoc user enquiries requiring timely informed response.
- (iii) Existence of many operations which requires continuous almost constant access to the communication channels. Resources cost to accomplish this requirement couldn't be justified.
- (iv) Reliability, to guarantee result in a competitive environment computing facilities must be available throughout.

Security: confidentiality, authenticity and secrecy of learner performance records should be certain. Others includes robustness, efficiency etc.

CHAPTER FIVE: METHODOLOGY

5.1 Introduction

In this chapter we provide a brief description of the working of AB3S and present an overview of agent-oriented analysis and design methodologies. It further details how the selected methodology is used to realize AB3S. To analyze, design and develop the envisaged system Agent oriented methodologies were used. According to [Opiyo et al paper] methodologies are bodies of methods employed by disciplines, while the methods are procedures for attaining objectives. The role of agent-oriented methodologies is to assist in all the phases of the life cycle of an agent-based application development, including its management. The GAIA methodology (Zambonelli et al, 2003) is described and used to analyze and design the AB3S. Other methodologies are used to address limitations of GAIA in development of AB3S.

5.2 Overview of the system

The envisaged agent –based service support system should enable users perform administrative duties efficiently, effectively and at minimal cost in a competitive, dynamic and distributed environment. The system should provide users with interface to capture and access details of courses, prospective and continuing students, tutor, administrator as well as other course administration information. The system should be open and accessible in a distributed environment.

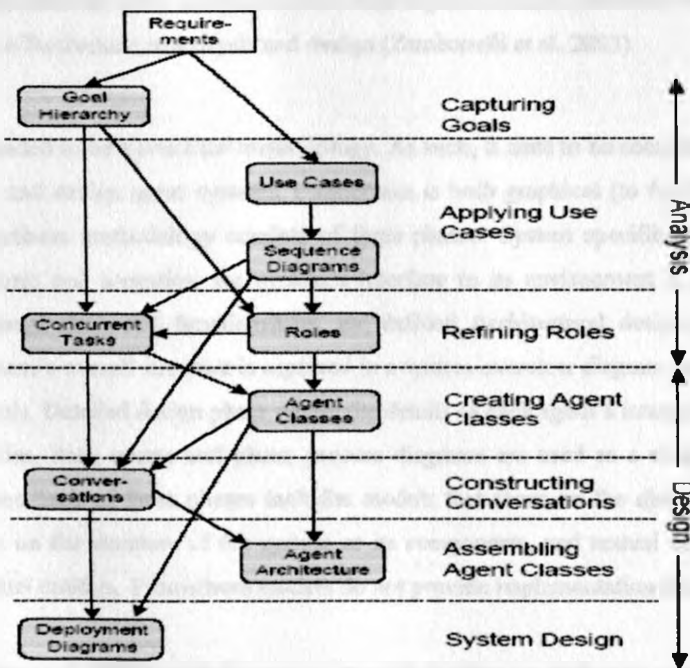
The system provides users with interface to configure software agent in order to specify appropriate correspondence period. The system also holds data in a persistent database which serve as input and output reference point for the agents. The system provides communication channels to the agents to enable them relay information they process to intended recipients as scheduled. The system should provide interface to deploy agent in their execution environment to enable them perform intended functions. Once deployed software agents should work autonomously and offload users of routine and laborious work. Agents' should submit result of their operation by updating database and or generating summaries.

The system should generate summaries and avail the same to intended recipients using appropriate communication media. System users should be able to obtain summaries either by logging into the system or through their mobile phone and or their email accounts.

5.3 Agent Oriented Analysis and design

The effort to reduce time and cost in developing automated systems leads to the search of efficient methodologies for system development. There exist a number of agent-oriented analysis and design methodologies in the market. Various methodologies apply to different stages of system development differently. Among the agents' methodologies are *Multi-agent Software Engineering* (MaSE), SADAAM (Deloach,2001), Tropos, (Giorgini et al. 2003), *Prometheus* (Padgham & Winikoff 2002) and *GALA* (Zambonelli et al. 2003).

The Multiagent System Engineering (MaSE) methodology, takes an initial system specification, and produces a set of formal design documents in a graphically based style. The primary focus of MaSE is to guide a designer through the software lifecycle from a project specification to an implemented agent system. MaSE is independent of a particular multi agent system architecture, programming language, or message-passing system. A system designed in MaSE could be implemented in several different ways from the same design. MaSE also offers the ability to track changes throughout the process. Every design object can be traced forward or backward through the different phases of the methodology and their corresponding constructs. An overview of the methodology and models is shown figure 5.1 below.



. The MaSE Methodology

Figure 5.1 MaSe Methodology

The methodology is however suitable for closed domain systems (Tveit 2001). MaSE has no organizational abstractions other than role models; it has no organizational rules or organizational structure (Zambonelli et al. 2003)-making it unsuitable for open domains like Internet-based systems.

Tropos (Giorgini et al. 2003) covers all phases of the software development: Early Requirements, concerned with the understanding of a problem by studying an existing organizational setting; the output of this phase is an organizational model, which includes relevant actors and their respective dependencies; Late requirements phase, where the system-to-be is described within its operational environment, along with relevant functions and qualities; this description models the system as a (small) number of actors which have a number of dependencies with actors in their environment; these dependencies define the system's functional and non-functional requirements; Architectural design phase is where the system's global architecture is defined in terms of subsystems, interconnected through data and control flows; within the framework, subsystems are represented as actors and data/control interconnections are represented as (system) actor dependencies; Detailed design phase is where each architectural component is defined in further detail in terms of inputs, outputs, control, and other relevant information.

Tropos offers in comparison with other existing methodologies two main advantages. First, Tropos covers the early stages of requirements analysis, and thus allows for a deep understanding of not only the system itself, but also of the environment where the system will operate and also helps to better understand the interactions that will occur in the system between the software agents and the humans. Second, *Tropos* covers the full range of the software development phases from the early analysis to the actual implementation. However does not use organizational rules in undermining its effectiveness in analysis and design (Zambonelli et al. 2003)

Prometheus is intended to be a *practical* methodology. As such, it aims to be complete: providing everything that is needed to specify and design agent systems. Prometheus is both graphical (to facilitate overview) and textual for details. The Prometheus methodology consists of three phases: System specification phase, where the system is specified using goals and scenarios; the system's interface to its environment is described in terms of actions, percepts and external data; and functionalities are defined Architectural design phase where agent types are identified; the system's overall structure is captured in a system overview diagram; and scenarios are developed into interaction protocols. Detailed design phase where the details of each agent's internals are developed and defined in terms of capabilities, data, events and plans; process diagrams are used as a stepping stone between interaction protocols and plans. Each of these phases includes models that focus on the *dynamics* of the system, (graphical) models that focus on the structure of the system or its components, and textual descriptor forms that provide the details for individual entities. Prometheus models do not provide implementation details (Chan et al. 2004).

GAIA (Wooldridge et al. 2000) methodology is intended to allow an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. GAIA uses the organizational metaphor to model multi agents System (MAS). The GAIA methodology generates a set of organizational abstractions which are necessary for designing and building systems in complex, open environments. According to GAIA, MAS is computational organization of agents, each agents playing a specific roles in the organization, and cooperating among them in order to achieve the organization goal. GAIA methodology considers analysis and design stages of system development and fails to consider the collection of requirements or systems implementation

5.4 Selected Methodology

The GAIA methodology is selected for the analysis and designs of the envisaged system because of the abstractions it uses (environmental model, organization rules and organizational structures). GAIA methodology is suitable for open environments such as AB3S which is to be open and is expected to run in an Internet-based environment. Further, the methodology is not tied to any implementation tool.

5.5 The GAIA Methodology

GAIA methodology considers analysis and design stages of system development and generates five models as shown in the figure 5.2 below.

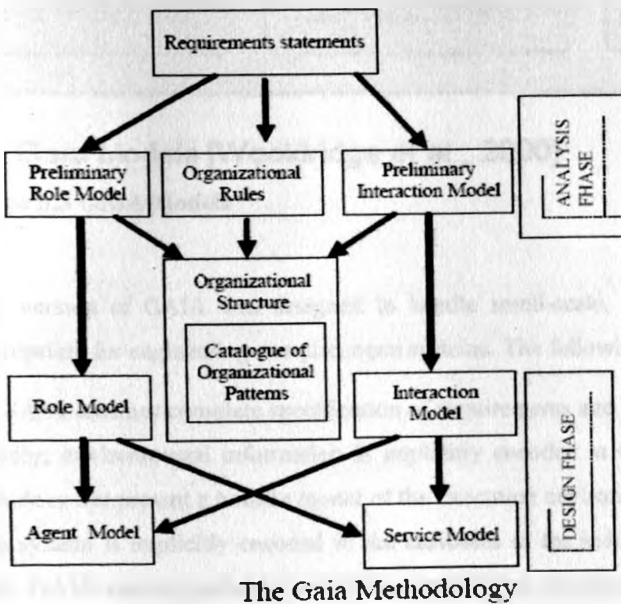


Figure 5.2 The GAIA Methodology

Using GAIA methodology five models are constructed which are role model and interaction model constructed during analysis stage while Agent model, service model and acquaintance model are constructed in design stage.

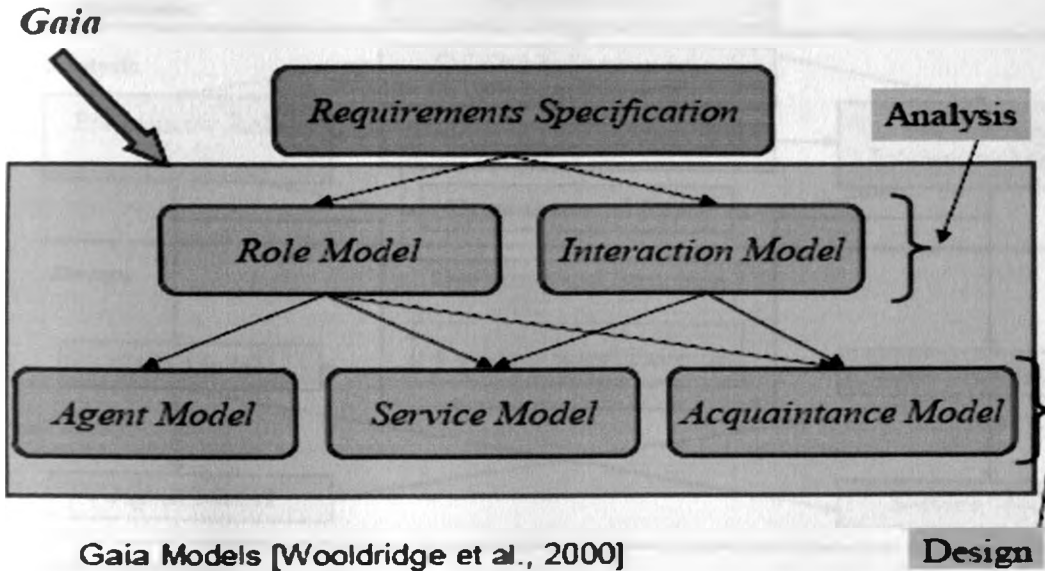


Figure 5.3 GAIA Models

Initial version of GAIA was designed to handle small-scale, closed systems. It has weaknesses that render it inappropriate for engineering complex open systems. The following weaknesses are noted.

First GAIA assumes complete specification of requirements and does not address the requirement-gathering phase. Secondly; environmental information is implicitly encoded in the permissions and protocols of individual roles. GAIA does not present a holistic model of the execution environment to the developers. Thirdly domain knowledge in the system is implicitly encoded in the attributes of the individual roles. Fourthly roles are not hierarchical in GAIA. GAIA cannot explicitly model the organization structure of the agents in the system, or alternatively, the architecture of the system. It also lacks the ability to explicitly model the social goals, social tasks or social laws within an organization of agents. Most of these initial GAIA limitations have been addressed some how in GAIA V.2 and ROADMAP(GAIA V.3). Improved GAIA is proper for the development of systems that work in complex and open environment.

5.5.1 GAIA Analysis

GAIA starts at the analysis phase; the objective of the analysis stage is to develop an understanding of the system and its structure. An organization is viewed as a collection of roles that stand in certain relationships to one another. GAIA analysis phase organizes collected specifications and requirements for the desired system into an environmental model, preliminary role and interaction models, and a set of organizational rules, for each of the sub-organizations composing the overall system as shown in figure 5.4

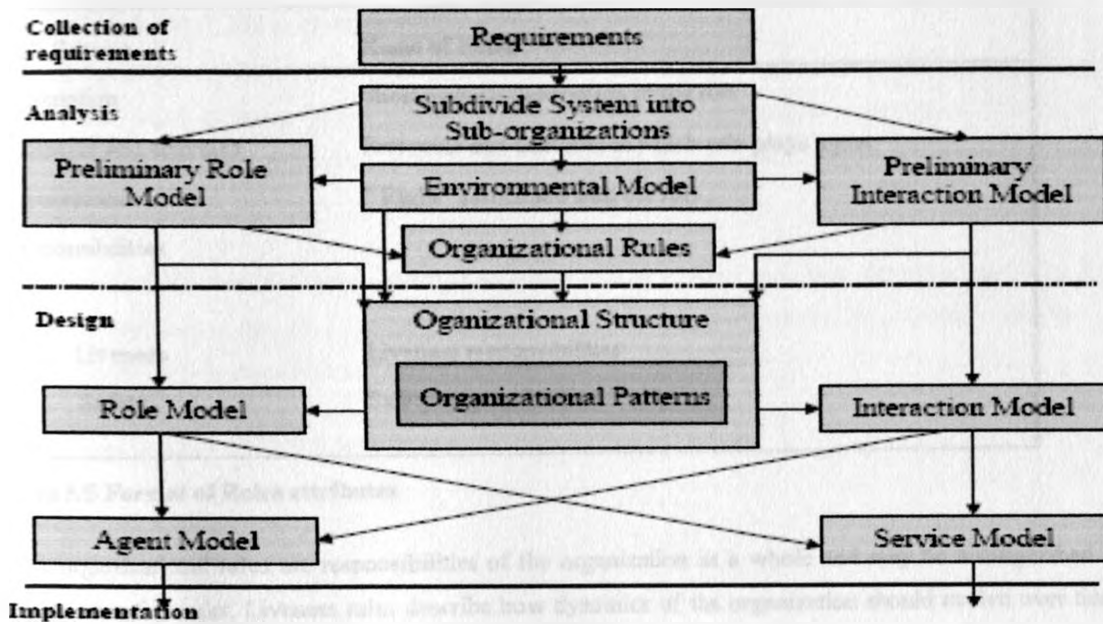


Figure 5.4 Detailed GAIA Model

The *environmental model* in GAIA is specified as a set of abstract computational resources such as variables or tuples that are made available to agents for *sensing*, *effecting* and *consuming*. GAIA uses a list of resources, each associated with a symbolic name, characterized by the type of actions that agents can perform on it. These environmental resources may be represented by *data structures* or *variables* at implementation phase. GAIA proposes that entities that act merely as data providers are modeled as resources while those capable of performing complex operations can be agentified.

The *preliminary roles model* identifies basic skills required by an organization to achieve its goals. These skills remain the same independent of the actual organizational structure. GAIA uses permissions and responsibilities to represent preliminary roles. Permissions relate agent roles to the environment in which they are situated and can also represent knowledge the agent can have. GAIA uses the same notation as that used to represent environmental resources, but the attributes associated with resources represent what agents playing such roles must be allowed to do to accomplish the role and what they must not be allowed to do. Responsibilities determine expected behavior of a role and are divided into *liveness* and *safety properties*. Liveness properties describe states of affairs that an agent must bring about, given certain conditions, while safety properties are aimed at maintaining acceptable states of affairs. GAIA specifies liveness properties using liveness expressions. GAIA makes up a set of role schemas as shown in figure 5.5

Role Schema	Name of Roles
Description	Short English description of the role
Protocols and activities	Protocols and activities in which role plays a part
Permissions	“ Right” associated with the role
Responsibilities	
Liveness	Liveness responsibilities
Safety	Safety responsibilities

Figure 5.5 Format of Roles attributes

GAIA organizational rules are responsibilities of the organization as a whole and may be distinguished as either *liveness* or *safety rules*. Liveness rules describe how dynamics of the organization should evolve over time; while safety rules define time-dependent global invariants for the organization that must be respected. Organizational rules can be specified using the same notation as for liveness and safety properties for roles. Liveness and safety organizational rules are specified separately for roles and protocols.

In summary the analysis stage of GAIA is completed by carrying out the following stages iteratively; one identify the roles in the system and producing a prototypical roles model; two, for each role identifies and document the associated protocols and produce interaction model ;three, stage one roles model is elaborated with roles permissions responsibilities, protocols and activity.

5.5.1 GAIA Design

The aim of a design phase is to transform the abstract models derived during the analysis stage into models at a sufficiently low level of abstraction that they can be easily implemented. The GAIA design process involves generating a number of models.

GAIA detailed design identifies the agent and the services models to act as actual implementations of agents and their activities. Agent model definition entails identifying which agent types are to be defined to play specific roles and how many instances of each type have to be instantiated in the actual system.

The services model identifies the main services that are required to realize the agent’s role. While the acquaintance model documents the lines of communication between the different agents,using the interaction model and agent model.

5.6 Applying GAIA in Analysis and dcsign phase

GAIA methodology was applied in the analysis and design of AB3S.

5.6.1 Applying GAIA in Analysis

Using GAIA analysis phase organization model was created. The approach used, capture system from systems' organizations point of view. System as an organization comprise of various roles which interact in away to accomplish organizations' goals. This phase resulted in creation of two models, roles and interaction models.

Roles models

The roles model identifies the key roles in the systems the following roles models were generated The following roles documents key roles occurring in the system, their permissions and responsibilities, together with the protocols and activities in which they participates. Various roles models were generated.

Role Schema : STUDENT (ST)

Description :

A person seeking course information for enrollment/registration

Protocols and Activities:

RequirementRequest, GiveRequirements

Permissions :

Generate Student details //owner of student information
Course requirements //owner of student requirements

Responsibilities

Liveness:

ST=(RequirementRequest, GiveRequirements)+

Safety :

- true

Figure 5.6 Student role

Course coordinator requests interested persons to register. Student make enquiries on courses offered and state their requirements on request.

Role Schema : REGISTRATIONHANDLER (RH)
Description : Receives application requests, vet every application, process and post successful applications.
Protocols and Activities: AwaitApplication, VetApplication, ProcessApplication, InformApplicant
Permissions : reads supplied applicant details // applicant personal and qualification details supplied Available course details // courses offered details Successful Applicants // application result or regrets generates successful applicant list
Responsibilities
Liveness: REGISTRATIONHANDLER -(all)* ALL -(AwaitApplication, VetApplication, ProcessApplication, InformApplicant)
Safety : • Applicant= registered student

Figure 5.7 Registration Handler roles

Applicants who are registered student submit courses application to registrar who evaluate it, record and post application results.

Role Schema : EXAMINATIONSCHEDULER(ES)
Description : Responsible for scheduling examinations. Accesses course registration information and generate schedule and post to centers
Protocols and Activities: <u>ReadCoursesApplications, GenerateExamSchedule, InformCenters</u>
Permissions : <p>Reads course requirement information // course information applied course information // list of requested courses</p> <p>Generates examination schedule.</p>
Responsibilities Liveness: EXAMINATIONSCHEDULER=(<u>ReadCoursesApplications, GeneratesExamSchedule, InformCenters</u>) ⁴⁴ Safety : *scheduled course=applied course

Figure 5.8 Examination scheduler role

Examination time table/schedule is generated based on course requirement and registration. A schedule is generated and posted to appropriate centers.

Role Schema : RESULTPROCESSOR(RP)
Description : Responsible for examination consolidation, moderation and generating a pass list. Result are posted to course Coordinator.
Protocols and Activities: <u>ReadCoursesApplications, ReceiveCenterResult, ConsolidateResult, ModerateResult, GeneratePassList, InformCenters</u>
Permissions : <p>Reads Centers' supplied students' examinations result // score information supplied registered course details // list of course applied</p> <p>Generates consolidated examination record. Pass list // student performance record</p> <p>Changes Pass list</p>
Responsibilities Liveness: RP =(RecurveCenterResult, <u>ConsolidateResult, ModerateResult, GeneratePassList, InformCenters</u>) ⁴⁴ Safety : *student examination result=registered course details *true

Figure 5.9 Result processor roles

Examination office obtains applied courses list, list of completed examinations from centers, consolidate and moderate. Pass list is generated and posted to appropriate centers.

Role Schema : COURSECOORDINATOR(CC)	
Description : Responsible for consolidation of marks from course instructors, submitting to the result processor and inform student of examination performance.	
Protocols and Activities: ReceiveExaminationResults,ConsolidateMarks,InformResultProcessor	
Permissions :	
Reads	instructor marks
	course applications
generate	centers' score record
Responsibilities	
Liveness: CC=(ReceiveExaminationResults,ConsolidateMarks,InformResultProcessor)+	
Safety : ●course results =registered Courses	

Figure 5.10 Course Coordinator roles

Course instructor read marks entered by instructor consolidate and inform result processor. Result processor role is played by registrar.



Figure 5.11 Registrar's responsibilities

Role Schema : COURSETutor(CT)	
Description :	
Responsible for entering student marks and informing course instructor	
Protocols and Activities:	
<u>RecordScore</u> , InformCoordinator	
Permissions :	
Reads	Supplied student details // access student details
register	Supplied registered course details // check whether student is registered from register
Changes	course score for registered student // record student score
Responsibilities	
Liveness:	
COURSEINSTRUCTOR=(RecordScore InformCoordinator) *	
Safety :	
•course score>=0 and course score<=100	

Figure 5.11 Course Instructor roles

Interaction Models

This model shows dependencies and relationships between various roles in a multi agent system, thus clearly show information flow within the system. Various interaction models were realized. one of the model is shown in figure 5.12 while the rest can be obtained from appendix C.

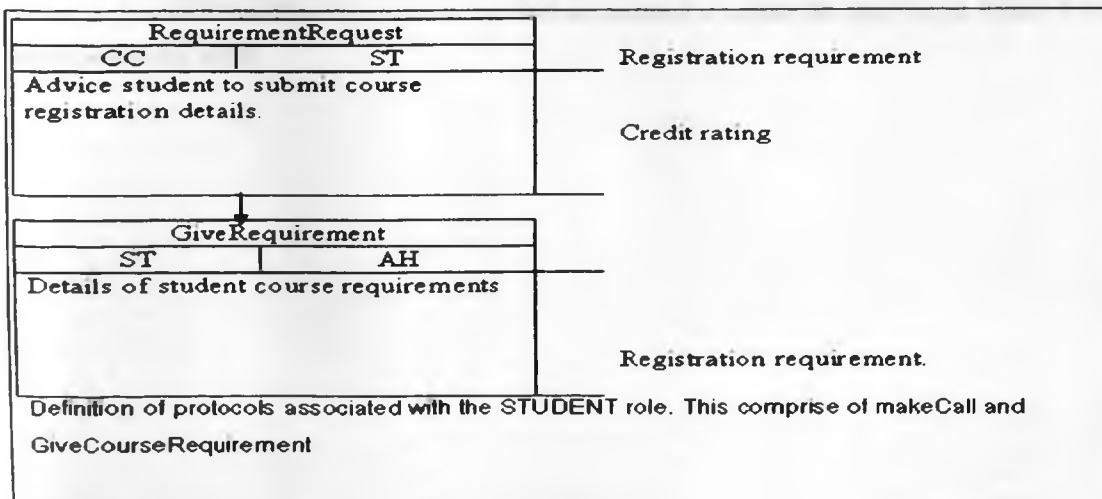


Figure 5.12 A sample Interaction model.

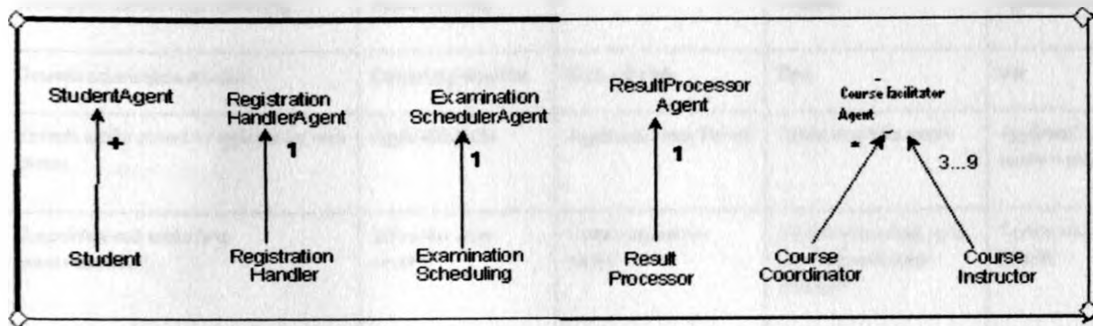
Applying GAIA in design

The GAIA design process involves generating three set of models, which are agent, service and acquaintance models.

5.6.2 GAIA Design Models

(a) Agent Model

Agent model definition entails identifying which agent types are to be defined to play specific roles and how many instances of each type have to be instantiated in the actual system GAIA detailed design identifies the agent and the services models to act as actual implementations of agents and their activities.



The Agent Model

Legend
+ : one to many instances
* : zero to many instances
1 : only one occurrences
n.m : n to m occurrences

Figure 5.13 The Agent Model

(b) Service model

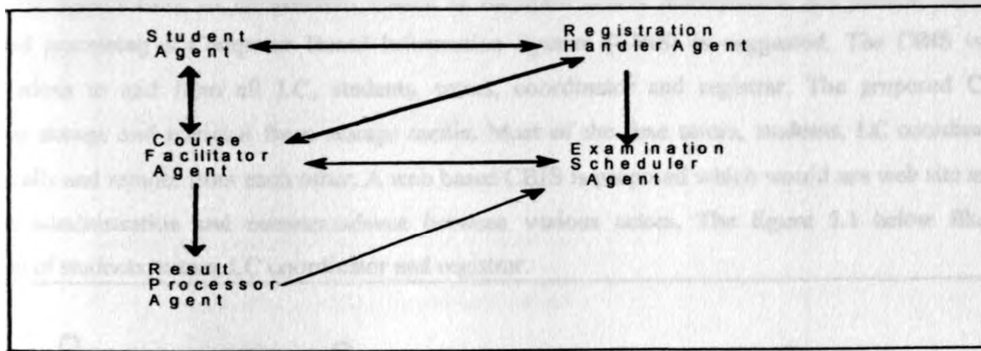
The services model identifies the main services that are required to realize the agent's role. Figure 5.13 represent service model for AB3S.

Service	Inputs	Outputs	Pre-condition	Post-condition
Obtain students Course Requirements	Applicant course details	Applicant course requirement	Applicant must be a registered student	true
Verify students' course Application	Applicant requirement	Valid and Invalid applications	Correctly filled application form	true
Process Application	Course requirement Valid applications	Successful and unsuccessful applications	Valid applications and course requirement available	Registered Course Applicant & registered student
Get courses application records From applicants for exam scheduling	Registered courses, classes schedule	Exam schedule.	Registered courses list available	#Courses schedule <= # registered courses.
Generate examination schedule	Courses registered for	Exam schedule	True	true
Records marks scored by applicant for each course.	Applicants marks	Applicants course Record	Course score= for course	Applicants' registered course = score
Consolidate unit marks from course instructor.	Instructors score records	Center examination record	All instructor submit result record for each course examined	Centers examination records.
Consolidate examination result from all Centers.	Marks from all centers	Single result record	All course units received	Consolidated result record
Moderate examination result	Consolidated marks	Final examination result record	All results received	Moderated results
Generate examination pass list for centers	Moderated results	Pass, deferred and fail list	True	Applicant performance records.

Figure 5.14 The Services model

(c) Acquaintance model

The acquaintance model documents the lines of communication between the different agents, using the interaction model and agent mode



The Acquaintance Model

Figure 5.15 The Acquaintance Model

5.8 Summary

In this chapter we were able to use the GAIA methodology to specify analysis and design models that are used in implementing AB3S in the next chapter. Agent models identify Agents which would be implemented to realize AB3S.

CHAPTER SIX: DESIGN

6.1 Introduction

Currently IU doesn't have an information system to facilitate course management and student support. To address data storage and processing a Computer Based Information System (CBIS) is suggested. The CBIS is to record data and communications to and from all LC, students, tutors, coordinator and registrar. The proposed CBIS would enhance information storage and retrieval from storage media. Most of the time tutors, students, LC coordinator and registrar are geographically and remote from each other. A web based CBIS is proposed which would use web site as a common interface for course administration and correspondence between various actors. The figure 5.1 below illustrates the expected interactions of students, tutors, LC coordinator and registrar.

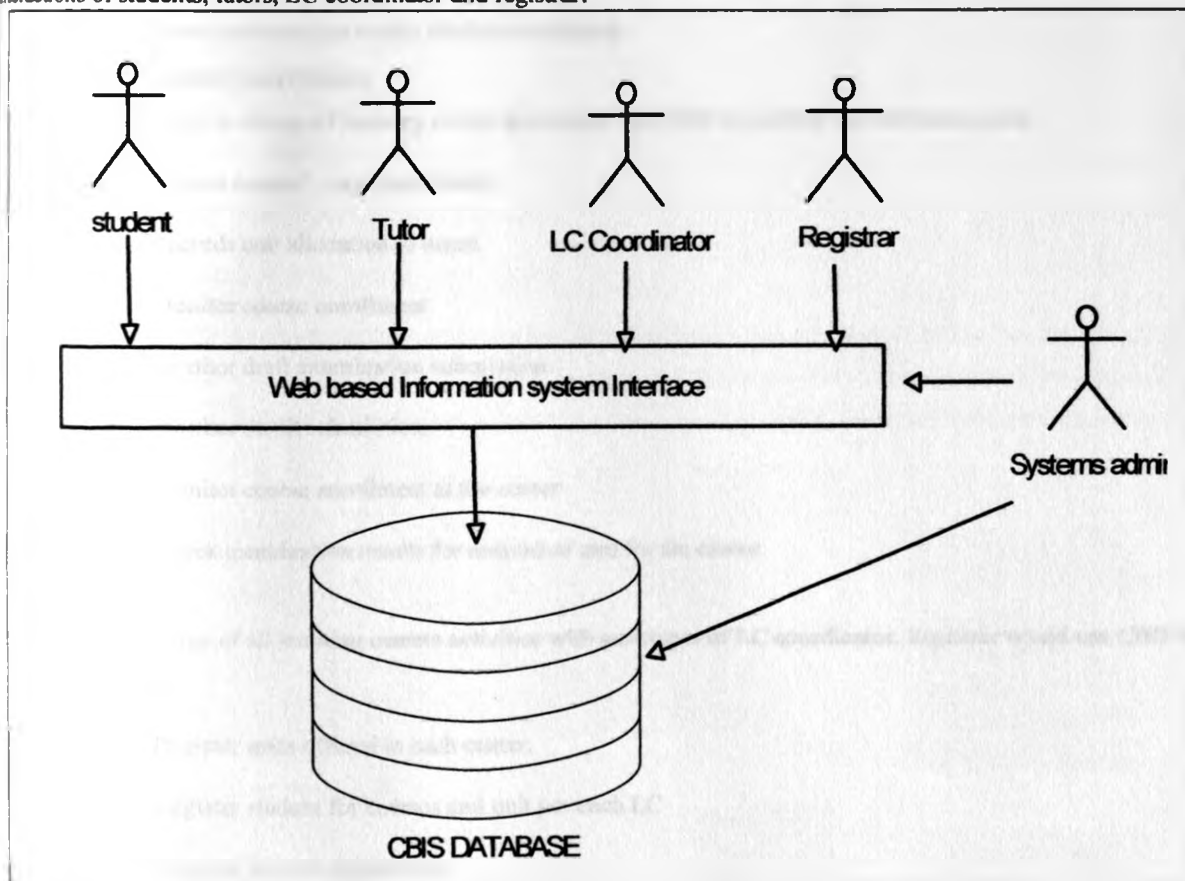


Figure 6.1 Interaction in Web Based System

6.2 Systems user

CBIS is expected to have many users each with a distinct role to play as specified below.

6.2.1 Student

Every student is expected to perform the following task using CBIS

- Check course registration requirement details
- Provide all required details during registration.
- Register for unit on or before due date

- Check course timings
- Check examination results.

6.2.2 Tutor

Tutor facilitates unit(s) allocated by teaching and evaluating. CBIS support the role of tutor. Tutor is expected to use CBIS to perform the following task.

- Check on units schedule and allocation
- Submit draft examination
- Record examination results for unit facilitated.
- Check examination results for unit facilitated.

6.2.3 Learning center coordinator

This is the individual in charge of learning center and would use CBIS to perform the following tasks

- Record centers' employee details
- Records unit allocation to tutors
- Monitor course enrollment
- Monitor draft examination submission.
- Monitor result submission.
- Monitor course enrollment at the center
- Check examination results for individual and for the center.

6.2.4 Registrar

Registrar is in charge of all learning centers activities with assistance of LC coordinator. Registrar would use CBIS to carry out the following.

- Register units offered in each center.
- Register student for courses and unit per each LC
- Monitor student registration
- Monitor unit allocation to tutor
- Schedule units for all LCs
- Monitor course progress
- Monitor tutor performance
- Generate/draft/post all correspondences to and from registrar office and student, tutor, LC and customer.
- Monitor draft examination submission

- Moderate examination results
- Post examination results to student ,LCC and Tutor
- Any other administrative role.

6.2.5 Systems Administrator

This would be individual super user in charge of overall systems and expected to ensure availability, reliability and integrity of the system. The administrator would be expected to do the following.

- Define users role in the systems
- Grant and or revoke user rights on the system
- Configure the system.
- Trouble shooting and recovery
- Back up and maintenance of the same
- Enhance overall system performance.

To realize the system the data should be stored in a data base which could be distributed to improve on data access and minimize network traffic. Class diagram in figure 5.2 below describe classes of the system required to realize the systems' data base. Various classes identified maps into tables which could be implemented using any relational DBMS.

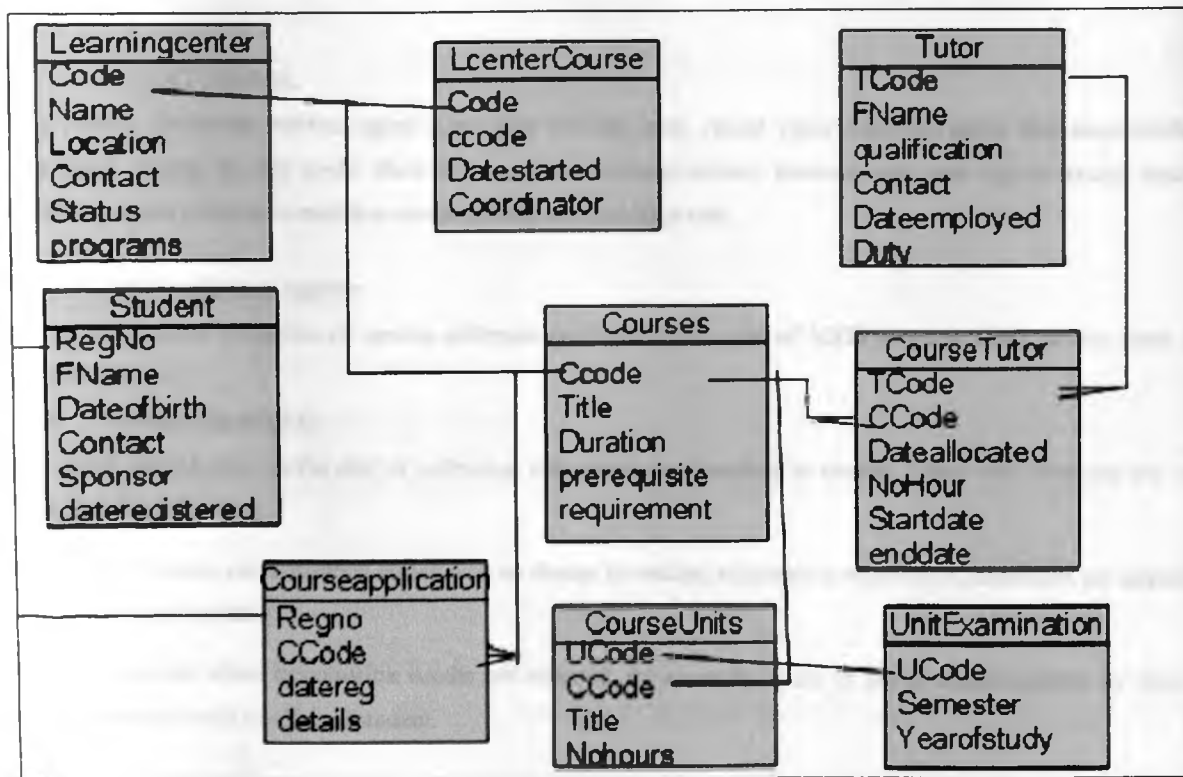


Figure 6.2 Basic database table design

From the proposed data base design it's anticipated that a comprehensive user interface would be needed to facilitate data capture and retrieval. The server hosting the web site and the database would contain a lot of useful data about students, applicants, tutor, examinations, performance, student progress etc. various reports based on the data can be generated to support decision making.

6.3 SOFTWARE AGENTS

6.3.1 Introduction

To support the course and acquire useful information from the distributed system environment, user would be required to initiate all information retrieval tasks and interact with the replies from the system directly. Initiating all tasks, monitoring and processing all responses would be time consuming, some time outdated and labor-intensive. These means that reaction time and decision making process is affected. For example to check and validate a single student claim on missing marks, registrar would be required to visit many web pages on the web based CBIS ,make frequent calls inconveniencing many while delaying decisions. Employing human personal assistances to interact with CBIS is one option, but this introduces the same human problems of inefficiencies and time dependency associated with manual information system. Creating software agents to act as users' assistance in CBIS is proposed. The proposed software agents would offload routine and laborious works from CBIS users while the effort can be expended on intelligent tasks. The software agents won't replace human user but support them by offloading some tasks.

6.3.2 AGENT MODEL

This model documents various agent types that will be used. Agent types represent agent that accomplishes associated role(s). In this model there is one to one correspondence between roles and agents except course facilitator agent which accomplishes course coordinator and tutor role.

6.4 AB3S software agents

The system would comprise of various software agents to assist user of AB3S in performing various tasks as specified below.

6.4.1 Student Agent (SA)

The agent should take up the role of collecting information and posting to student. Other roles includes but not limited to the following

- Monitor course administration issues such as change in content, registration requirement, schedules etc and alert student appropriately.
- Alert student when examination results are released, for example incase of fail or supplementary an alert is immediately sent to affected student.

- Alert student of examination, schedule changes etc.

6.4.2 Tutor Agent (TA)

The agent should perform the following tasks on behalf of the tutor.

- Regularly monitor units allocation and send notification to concern tutor.
- Check unit(s) schedules and send reminder
- Play a general informative role.

6.4.3 Learning Centre Coordinator Agent (LCCA)

The agent would support LC coordinator in basic administrative issues by collecting and dispersing relevant information needed for decision making. The Agent would perform the following tasks on behalf of LC coordinator

- Alert tutors who have been allocated units to prepare for the same.
- Send memos to tutors and students on various course/units issues.
- Monitor course registration for the center
- Alert LC coordinator on
 - Examination setting progress.
 - Draft examination submission
 - Release of examination results.
 - Prepare unit allocation list per learning center and post to registrar.

6.4.4 Registrar agent (RA)

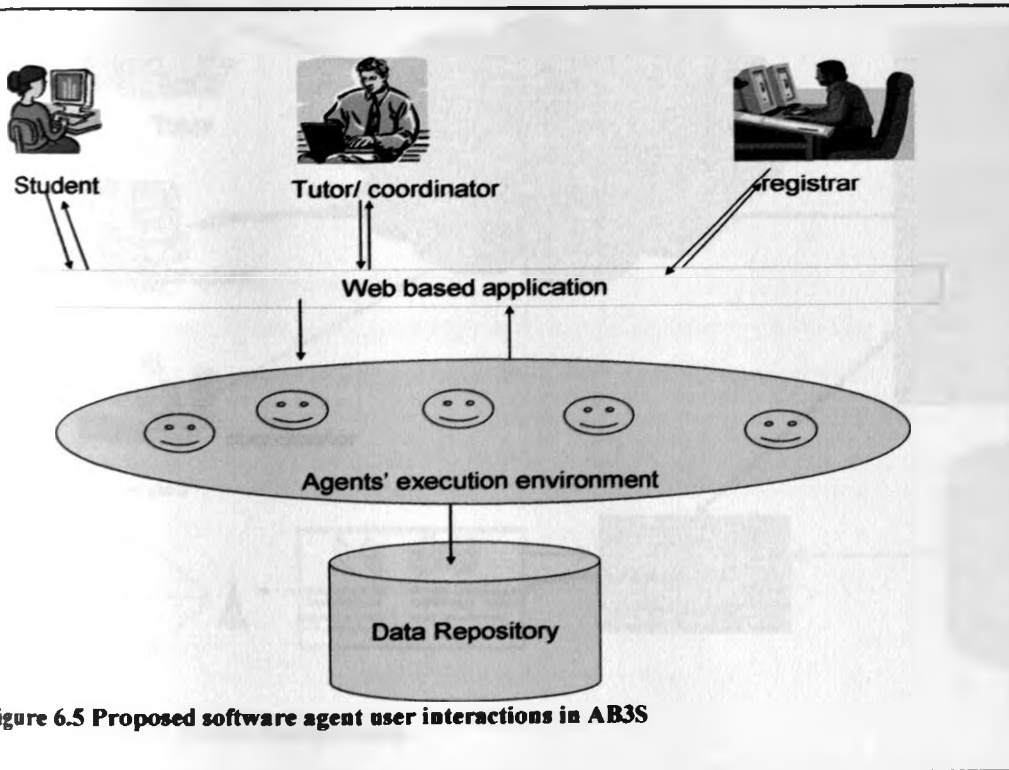
Registrar Agent would support registrar by offloading the following tasks from her.

- Monitor courses application from all centers on regular basis.
- Alert successful applicants.
- Monitor unit allocation to tutor from leaning centers
- Alert registrar on completion of
 - Draft examination submission
 - Submission of examination results
- Generate/draft/post some correspondences to and from registrar office ,student, tutor, LC and customer.
- Consolidate examination results from all LC.
- Post examination results to student ,LC coordinator and Tutor

To support various agents in the system, the software agent should be able to access system database, processing some data before posting findings to recipients' email box or text messages. Since content of database might exist in

remote site, the proposed system would comprise of mobile software agents to extract data at remote site, post to receipts on the site and submit required data to central database.

The figure below shows extension of web based CBIS to incorporate the software agents.

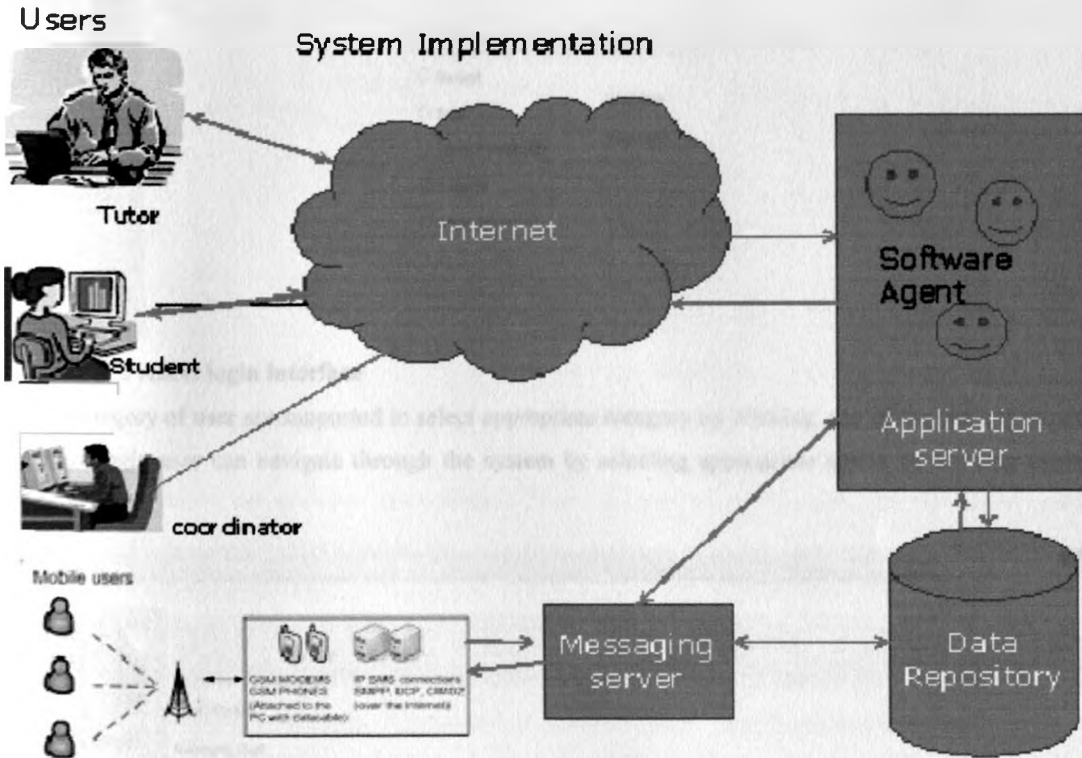


3 Figure 6.5 Proposed software agent user interactions in AB3S

CHAPTER SEVEN : IMPLEMENTATION

7.1 Introduction

This chapter describes how the AB3S prototype was developed and deployed for testing. The diagram below represents the overview of AB3S



4 Figure 7.1 System configuration

7.2 Data repository

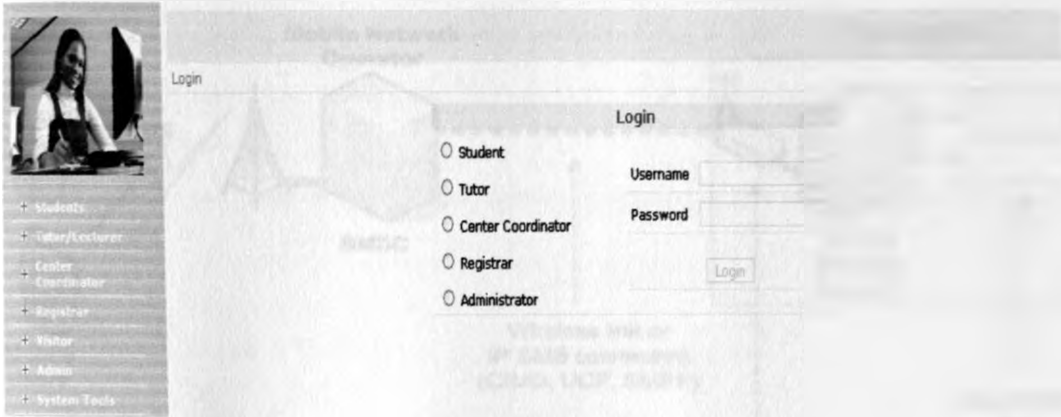
In order to implement a persistent data repository a relational data base was created using MySQL 3.7.1 . The database contained all the tables as designed and others for control and sequencing.

7.3 User interface

The AB3S require various user interaction, this is to capture user data and to relay information to users. To input data a comprehensive and interactive web page was developed using PHP,HTML and java script. Web page was designed using ADOPE Dream weaver CS3.To serve web pages apache web server was used. Various AB3S user can access the system using any web browser.

The system requires frequent and time dependent information exchange to and from user. Information from the system is sent to user through various means.

- User can interact with the AB3S directly by logging into AB3S web pages, this is dependent on provided user level.. A login form shown in fig 7.1 below is displayed once the user types correct URL from any browser.



5 Figure 7.2 AB3S login interface

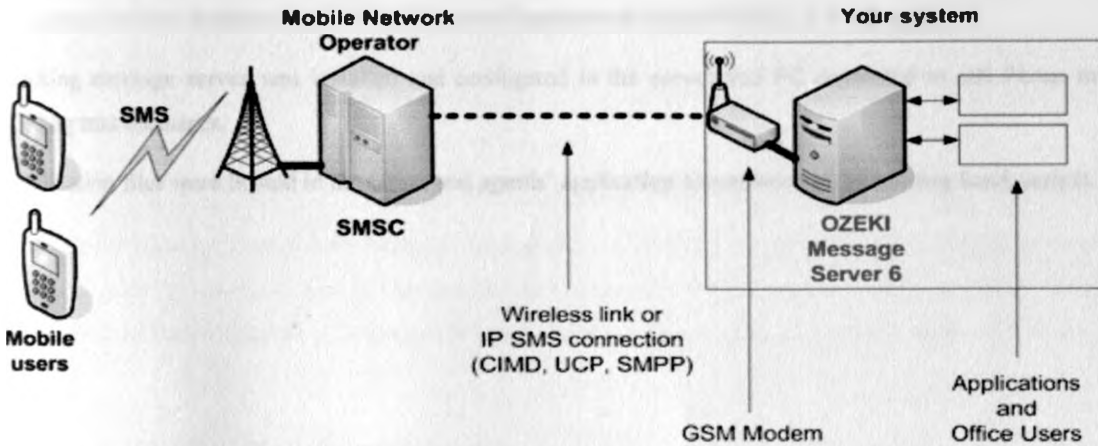
Various category of user are supported to select appropriate category by clicking one of the options provided. After successful login user can navigate through the system by selecting appropriate option on the user menu as show below.



6 Figure 7.3 center coordinator menu

- (a) To enhance information distribution, structured information can be relayed to specific user email address, at time some user may ignore AB3S, but not their email. To support emailing with AB3S mail server component is required; mercury mail server is used for these. Uses of email enhance system reliability by ensuring receipts get information.

(b) Small and urgent pieces of information can be relayed instantly to user by sending SMS to their mobile phone. The system is connected such that information recipient on mobile network can receive and respond to correspondences instantly. The figure below show how mobile phone is connected to the application.



7 Figure 7.4 phone to GSM Connection[courtesy of ozeki informatics]

Ozeki message server 6.0 provides appropriate API to enhance system functionality. By use of mobile phone system reliability and flexibility are enhanced.

7.4 User -agent interaction

Software agent work in system environment to perform some tasks on behalf of the user, in this work agents plays processing and informative roles. Agent access database, gather data ,process into required information and relay the same to user. Different users needs information at different times hence the need to interact with agents by configuring alert period. The figure 7.4 below shows agents configuration interface. Agent configuration ensures users are not bombarded with many alert unnecessary.

Agent Configuration

Indicate Durabon of E-mail alert.

Registrar Alert

Daily

Weekly

Monthly

Specify Other _____ Hours

8 Figure 7.5 Agent configuration interface

7.5 Deployment

To deploy AB3S ,XAMPP was used. XAMPP is application development and deployment tools consisting of phpMyAdmin for MySQL administration; Webalizer; php switch for development; mercury mail which is an inbuilt mail server and fileZilaFTP for file upload.

To connect MySQL database engine with the rest of application MySQLODBC3.5.1 was used.

Ozking message server was installed and configured in the server and PC connected to cell Phone modem for sending text messages.

Application files were hosted in the server and agents' application commissioned by running batch scripts.

CHAPTER EIGHT: ANALYSIS OF RESULTS

8.1 Running AB3S

To demonstrating that the AB3S prototype and its components are working according to their specification and that functional and non-functional requirements are satisfied as much as possible there was need to thoroughly test the AB3S.

To run AB3S for testing, we set up a web server using apache which is a component of Xampp.MySql ,mercury and apache services were started from Xampps' control panel. Ozeking SMS gateway was connected to a test database running in MySql database, and to the mobile service provider and connection tested. Registrar, tutor, student, coordinator and administrator software agents were placed in web server and launched using command scripts.

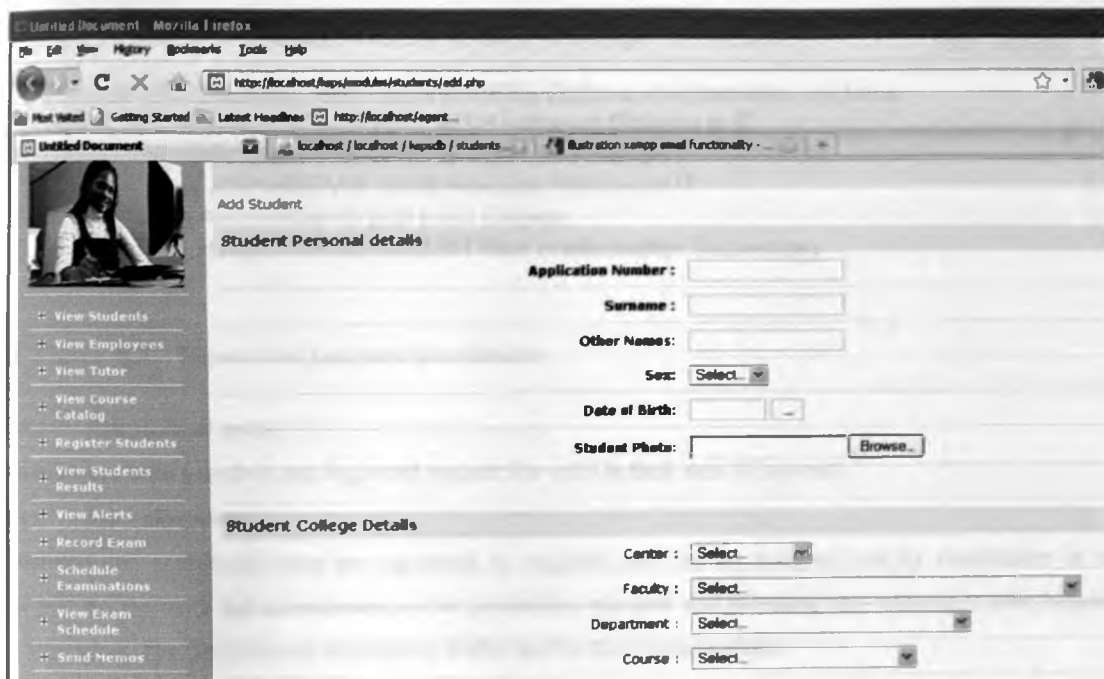
8.2 Testing

To test AB3S the following testing goals are used:

- Test whether the prototype system is running
- Test whether application is working without use of agents.
- Test interoperability of various software components used
- Testing ability of agents to identify information recipient and relay appropriate information to them.
- Testing the ability of agent system to facilitate administrative work.
- Testing ability of agent to offload routine information processing from human user.

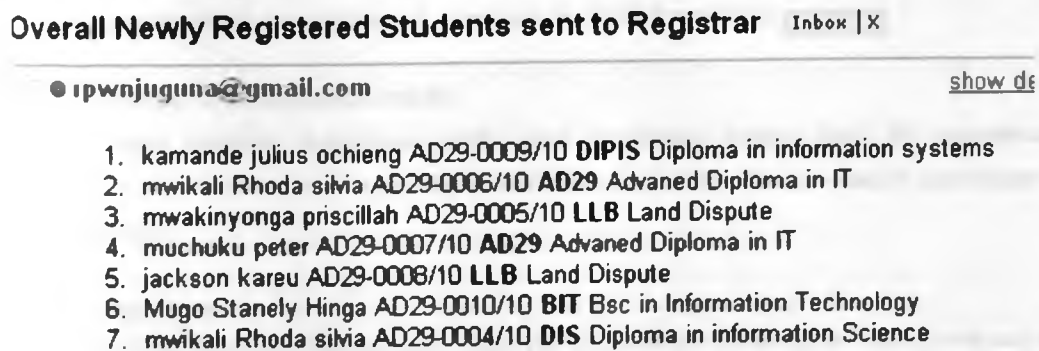
8.3 Registering student

Prospective student submit duly filled application form to center coordinator or to registrar. Official application form can be downloaded from IU main web site. Coordinator and or registrar login into AB3S and access online request form and key in details.



9 Figure 8.1 Student application form

Registrar access AB3S to perusal through the list of student and can view new application which she may decide to approve or reject based on admission criteria. If approved applicant is assigned registration number, an email and a text message is sent to successful applicant instantly. Registrar agent access database identifies list of successful applicant and send a summary to center coordinator and full list to registrar. Sample report to is shown in fig 8.2 and 8.3 below



10 Figure 8.2 Report generate by agent

Respective center coordinator receive agents' filtered report as shown below in fig 8.3

1. kamande julius ochieng AD29-0009/10 DIPIS Diploma in information systems
2. mwikali Rhoda silvia AD29-0006/10 AD29 Advaned Diploma in IT
3. mwakinyonga priscillah AD29-0005/10 LLB Land Dispute
4. muchuku peter AD29-0007/10 AD29 Advaned Diploma in IT
5. jackson kareu AD29-0008/10 LLB Land Dispute
6. Mugo Stanely Hinga AD29-0010/10 BIT Bsc in Information Technology

11 Figure 8.3 Report sent to center coordinator

8.1.1 Selection of units

Upon registration a student can login and register for units in their area of interest.

8.1.2 Unit allocation.

Qualified and scrutinized tutor are registered by registrar and can be assigned unit by coordinator in respective centers. To facilitate fast communication for immediate response and planning text message is sent informing tutor of allocation. The detail course description is also sent to their email address.

Registered tutor can also login into AB3S and learn of allocation.

8.1.3 Registration of special/supplementary examination

Student who fail to satisfy examiner may apply for special or supplementary examination using AB3S.

Submission of draft examination

Tutor facilitating a particular unit prepare and submit draft examination to registrar

8.3 Scheduling of examination

Examination are scheduled and this served as basis for sending of memos and other notification by the software agents. Agent compared current date and scheduled one to send reminders.

8.4 Release of examination result.

Registrar upon receiving examination results from coordinator counter check for correctness, validity and consistence and release result upon which result are relayed to affected student ,coordinator and tutor for correspondence.

8.5 Summary

In general it was witnessed that the AB3S prototype demonstrates that an agent-based system can be implemented for course administration that allows agents to offload routine and laborious tasks of information handling from human user. The user from registrars' office were satisfied and also recommended additional functionality, which can be later implemented. The agents in AB3S were able to engage in multiple information role that lead to timely correspondence .

CHAPTER NINE : CONCLUSION AND RECOMMENDATION

9.1 Introduction

This part evaluate research finding and evaluate them against project objectives stated during proposal, the section also outline challenges encountered, lesson learned and recommendations.

9.2 Evaluations of objectives

This part discuss the extend to which project objectives were achieved as detailed below.

Objective 1 *To research on agent based technology and applications and utilization of software agents in the area of education and training*

This objective was achieved as anticipated, we established that agent technology present many opportunities as well as challenges and use of agents in application is justifiable.

Objective 2 *Examine and evaluate existing communication technologies that support distributed organisations such as institutions, banks, retail enterprises etc.*

We were able to meet this objective as we established various communication mechanisms which differs in terms of availability, applicability and suitability, It was noted that blend of communication methods can be used to support communication needs of organization.

Objective 3 *Examine tasks that could be offloaded from human users and assigned to software agents.*

This objective was achieved. We established that some of tasks performed by users are routine and time consuming; activities such as sorting out applications, informing various users could be automated by offloading them to software agents. We developed AB3S which consist of registrar agent ,coordinator agents and others which process information from the database and communicate their finding/results to appropriate user using various communication technology.

Objective 4 *To propose a model for implementing an AB3S for medium to large size organization.*

We were able to specify a model for implementing AB3S.This model is in line with open system where agents are plug –in hence new agents with new features can be incorporated to enhance system functionality.

Objective 5 *To implement an agent based system based on the above model.*

We designed and successfully developed an agent based system(AB3S) with several agents to carry out varied activities in course administration. The agents' tasks implemented ranged from identifying information recipients ,processing information and communicating finding to various recipients. The role of software agents in the AB3S is simply supportive, some tasks mainly communication was delegated to software agents. Users of AB3S can obtain information within LAN as will as in the Internet.

Objective 6 *To test and evaluate the prototype using suitable test cases*

This objective was met, the implemented AB3S prototype was found to perform the intended tasks and thus it is a proof that agent can be built to support course administration functions. The system was found to achieve most of the functionality proposed in the analysis as described below:

The following were achieved satisfactorily:

- AB3S uses a web based interface to provide users with facilities to capture user as well as client details for submission to database for agents processing.
- AB3S used modern and effective communication mechanisms such as emails among many others ,this improved on system dependability.
- Support course administration function by delegating routine and laborious tasks to agents.
- It generates application summaries and communicates to concern parties hence improving on response time.
- Generates and post reminders to course administrators.

Beside the above stated functional requirements AB3S achieved other non functional objectives as stated below

- **Security:** Users of AB3S are allowed access to limited features just enough for performing their tasks. The restricted access ensure data integrity and validity of data into or out of the system.
- **Availability :** AB3S is hosted on a server and every user with access permission can access it any where any time. AB3S ensure minimal interruptions incase of upgrading.
- **Reliability:**By use of alternative or combination of communication methods , relay of information is guaranteed as failure of one wont affect the other.
- **Scalability :**Agents are implemented as stand alone entities with minimal interactions, hence agents services can be enhanced with minimal service interruptions.

9.3 Challenges

Despite achieving stated objectives numbers of challenges were encountered. The project was challenging and a lot of time and effort was used in order to meet its objective. Much time was spent on evaluation of objectives and definition of system boundary. Initially a blurred distinction existed between automated systems and agent based one. Much time was also spent identifying and evaluating agent's tools. It emerged that development tools were either not available, contained bugs, platform dependent or not compatible with other tools. Much time was spent on learning new development tools which frequently changed.

Another problem experienced was lack of a single clear development methodology or compatible ones, for the entire process hence the need to blend another of them which further complicated the process.

Another challenge was lack of agent literature. Agents' technologies are relatively new and have attracted a lot of interest, hence existence of few subjects' complete books or write ups. The much interest has led to existence of scattered contents of which some were not authentic. The focus has been on some aspects of agent while others are ignored.

The other challenge was dependent on external service provider, to test agent's functionality a reliable communication was needed which at time was not dependable. Another challenge was in testing, to investigate agents' performance, a time dependent test was required some running into weeks hence consuming a lot of time.

9.4 Lessons learned

I learnt a lot during the course of this project. I gathered a lot about agent technology and its diverse application areas. I learnt that agent technology is relatively new and upcoming field of computing, there is a lot of research going on and more need to be done to fully realize agents' potential. Adoption of agent technology wouldn't replace existing systems, but improve on them and revolutionize new ones.

The frequent change of identified tools enabled me to sharpen my programming skills especially on web based development. To realize AB3S various tools were used which required a great deal on configuration, I gained a lot in this area. These research work required performance of various tasks in parallel as well as repeating some, these required patience and time management these are life support skills I augmented. The frequent changes of research topic helped to sharpen my research skills.

9.5 Recommendation

Based on my experience during the research, I would like to recommend the following:

a) Changing in student enrollment procedure.

Education managers should re evaluation student enrollment procedure, instead of quarterly based approach the process should be continuous. This will not only relieve system from high peaks but also give time for all players to plan.

b) Existence of common Standardized student database

A lot of time is spent on selecting qualified candidate, informing them who fails to turn up, but may reapply again. These lead to wastage in terms of time and resources. If addressed prospective student history can be gathered and used to automate admission and also support planning.

c) Integration of systems

Various systems exists but manages different aspects of education. These leads to overlap and duplication of data and efforts which is wasteful, inefficient and costly.

d) Tools support and Agent development methodologies

More research needs to be done to help develop and standardize tools used for agent development and to avail appropriate supporting literature and manuals. Similarly there is need to research and propose standard life cycle methodologies that can be used analyze, design, develop and deploy agent-based systems.

9.6 Further Development

AB3S can be further developed to support other administrative functions in distributed enterprises. To enhance AB3S performance, Agents in AB3S could be mobilized to enable migration to point of use; these would facilitate local processing and decongesting network.

Research need to be done on how agents could be hosted on the web page to facilitate user interaction to make agent more responsive. AB3S should be extended to support data push and pull model in both direction.

Agents in AB3S should be improved by adding intelligence facet, such agents would learn of user behavior, admission requirement and advice users appropriately. Another improvement could be in the area of agents

cooperation. AB3S agents' interaction is at point of data access, further research need to be done to establish how one agent could collaborate with another to enhance systems' functionality.

9.7 Conclusion

This study focused on possibility of employing agent technology as the way to improve human productivity by offloading routine and laborious tasks from them. From my research I can clearly state that agent technology have indeed attracted a lot of interest and with continued research the technology will revolutionize software industry. The phenomenal growth of Internet and the World Wide Web have prompted scientific communities to join effort in promoting agent technology and my work contribute toward unlocking agents' potential.

The research effort was centered on the course administration function of a tertiary training institution with branches distributed across wide geographical region. Its my hope that a complete version of AB3S would be used to support functions of a distributed business enterprises.

1. **Amir Zeid**, Key Components of Agent-Based Development (1999), Computer science Department, The American University in Cairo
2. **Aylett R., Brazier F., Jennings N.R., Luck M, Nwana H., Preist C.**, (1997), .Agent Systems and Applications., Second UK Workshop on Foundations of Multiagent Systems FoMAS.97)
3. Automated Negotiation: Prospects, Methods and Challenges, International Journal of Group Decision and Negotiation, GDN2000 Keynote Paper.
4. **Brian Bewington, Robert Gray, Katsuhiko Moiszumi.** David Kotz, George Cybenko, Danila Rus, Mobile Agents in Distributed Information retrieval Dartmouth College.
5. **C. Iglesias, M. Garijo, J.Gonzales.** A survey of agent-oriented methodologies(1999)
6. Cougar (2004), .The Power of Agent-Based Systems., A White Paper, Cougar Software Inc., http://www.cougarsoftware.com/Knowledge_Center/Documents/Power%20of%20Agents.pdf
7. **Chan K., Sterling L., Karunasekera S.** (2004), .Agent-Oriented Software Analysis.
8. **Deloach A.S.** (1999), Multiagent systems Engineering: A methodology and Language for Designing Agent Systems
9. **Elisha T.Opiyo, Erick Ayienga, Katherine Getao, Bernard Manderick, William Okello-Odongo, ann Nowe,** Modelling a Multi-Agent Systems Scheduler for Grid Computing Using the Gaia Methodology.
10. **Franklin S., and Graesser A.** (1996), .Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents., Proceedings of the Third Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag 1996
11. **Giorgini P., Kolp M., Mylopoulos J., and Pistone M.**, (2003), The Tropos Methodology: An Overview
12. **Grabner M., Gruber F., Klug N. L., Stockner W.** (2000), .Agent Technology: State of the Art., <http://dis.eafit.edu.co/cursos/st725/material/sistdist/EstadoArte.pdf>
Handler, J., (1999), .Making sense out of agents., IEEE Intelligent Systems
13. **G. Glass**, "ObjectSpace Voyager Core package Technical overview", Mobility ,process, computer and agents, (Addison Wesley 1998)
14. **Ghanea-Hercock, J collis, D Ndumu,** Co-operating Mobile Agents for distributed parallel processing, autonomous Agents(99)
15. **Hong Hong, Ashok patel** (2002), Adaptivity through the use of mobile agents in web based student modeling.
16. Hong Hong, Application of Mobile Agents in web-based Student Modelling(2000)
17. Intelligence, special issue on Intelligent Agents and Multi-Agent Systems
18. **Joseph Jesson**, The design and implementation of Services to Support a Mobile Agent Architecture.(2000)
19. **Jennings N, Wooldridge M** (1995) Applying agent technology. J. of Applied Artificial

- 20 **Jennings N.R., Faratin P., Lomuscio A.R., Parsons S., Sierra C., & Wooldridge M., (2000),**
- 21 **Jennings, N. R. and Wooldridge, M. (1998b) Applications of Intelligent Agents. In: Agent Technology: Foundations, Applications, and Markets, pp. 3-28.**
Juan C, Alvaro A., Jose P., (2004), Paving the Way for Implementing Multiagent Systems: Integrating GAIA with Agent-UML.
- 22 **Kurt Rothermel, Markus Schwehm, Mobile agents(1998)**
- 23 **Mihaela Dinsoreanu, Ioan Salomie, Kalman pusztai, On the Design of Agent-Based Systems Using UML and extensions**
- 24 **Mark F. Wod, Scott A. DeLoach An Overview of the Multiagent Systems Engineering Methodology(2000)**
- 25 **McDonald, J.T Talbert, ML, DeLoach, S.A Heterogeneous Database Integration Using Agent Oriented Information Systems, Proceedings of the International Conference on Artificial Intelligence(2000)**
- 26 **Nwana ,H.S Software Agents : an Overview , Knowledge Engineering Review 11(3); 205-244**
OMG, Unified modeling language specification(version 1.5) UML revision task Force(2002)
- 27 **Raouf Boutaba (2001), Process Migration and mobile agents in networks management**
- 28 **Tista Kapoor, Packer Ali Dhamim, Design and Implementation Aspects of Mobile Agents.(1999)**
- 29 **Walworth, M ,& Herrick, R,J(1991), The use of computers for educational and testing purposes, Proceeding Frontiers in education conference.**
- 30 **Wood M., DeLoach A.S.(2000), An overview of the Multiagent Systems Engineering Methodology**
- 31 **Wooldridge, M., (2002), *An Introduction to Multiagent Systems*, John Wiley and Sons Ltd**
- 32 **Wooldridge M, Jennings N.R., (1994), .Agent Theories, Architectures and Languages: A Survey., Workshop on Agent Theories, Architectures & Languages (ECAI'94**
- 33 **Wooldridge M., Jennings N.R., Kinny D., (2000), .The GAIA Methodology for Agent- Oriented Analysis and Design., *Autonomous Agents and Multi-Agent Systems*, 3, 285-312, 2000, Kluwer Academic Publishers, Netherlands.**
- 34 **Yariv Aridor and Danny B. Lange(1998). Agent Design Patterns: Elements of Agent Application Design**
- 35 **Yariv Aridor and Mitsuru Oshima, Infrastructure for mobile agents: requirement and design**

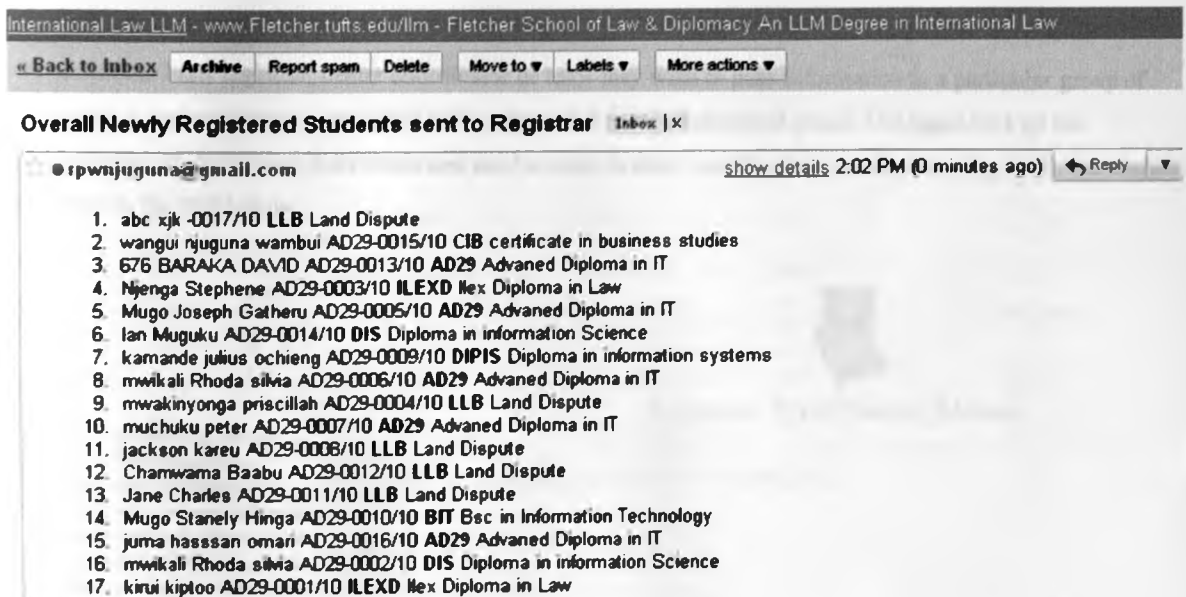
APPENDICES

Appendices A : Reports

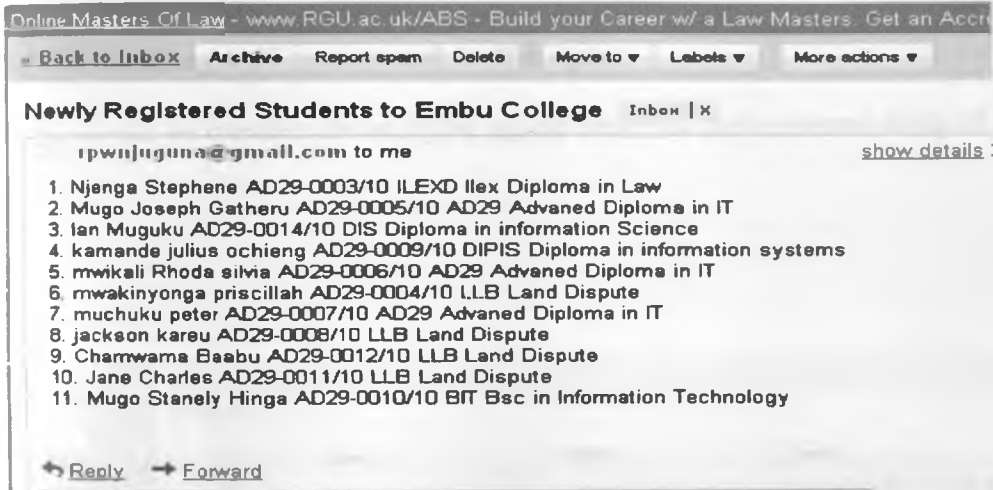
AB3S generates a number of reports for various users samples of such reports are as follows.

On daily basis registrar agents peruse through the database extract list of successful application and relay to

- a) Registrar a list of all registered student for the entire institution as shown in fig 10.1 below
- b) Center coordinator a list of student in that center as shown in fig 10.2 below

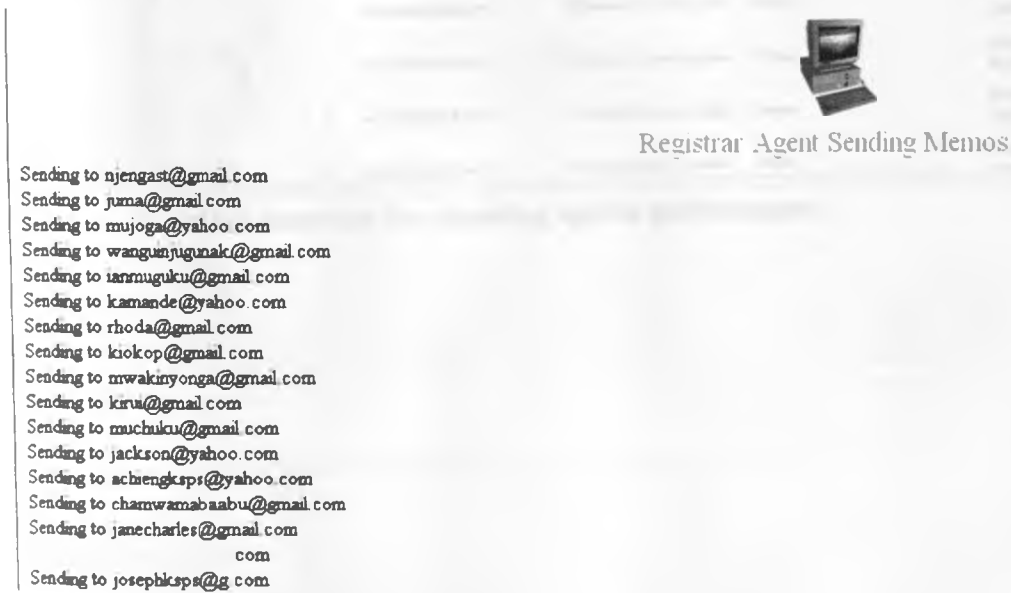


12 Figure 10.1 List of registered student



13 Figure 10.2 List of student registered in one of the centers.

On occasional basis registrar, center coordinator or tutor may wish to pass information to a particular group of recipients. A registrar agent is provided with content and name of recipient group. The agent take up the responsibility of identifying individuals and send content to their email account as shown in fig 10.3 . the content is as shown in fig 10.4 below.



14 Figure 10.3 Visual display of agent in action sending mails

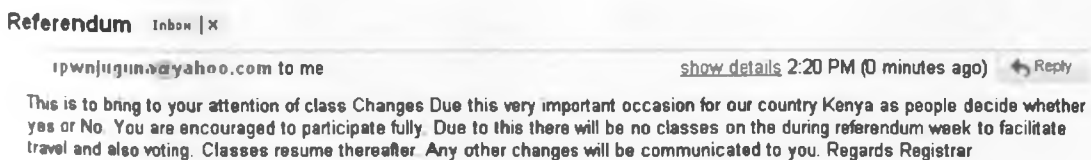


Fig 10.4 Sample email content sent by the agent.

AB3S provide an interface for verifying agents activities such as shown in fig 10.5 below.

The screenshot shows the 'Alert Module' interface. On the left is a sidebar with navigation links: View Students, View Employees, View Tutor, View Course Catalog, Capture Student Details, View Students Results, View Alerts, Record Exam, Schedule Examinations, View Exam Schedule, Send Memos, and Set Your Email Alert. The main area has search filters for 'Email:', 'Subject:', and 'Search'. Below these is a table with the following data:

#	Sent To	Sub ject	Communication	Time
1	Hemland@ksp.s.ac.ke	Email	Newly applied students	2010-02-16 04:10:39
2	Embu College@ksp.s.ac.ke	Email	Newly applied students	2010-02-16 04:10:55
3	Parklands@ksp.s.ac.ke	Email	Newly applied students	2010-02-16 04:10:52
4	mujoga@gmail.com	Change of opening date	Memo	2010-02-10 16:57:57
5	mujoga@gmail.com	Change of opening date	Memo	2010-02-10 16:57:55
6	mujoga@gmail.com	Change of opening date	Memo	2010-02-10 16:57:46
7	mujoga@gmail.com	Change of opening date	Memo	2010-02-10 16:57:38
8	mujoga@gmail.com	Change of opening date	Memo	2010-02-10 16:57:06
9	mujoga@gmail.com	Change of opening date	Memo	2010-02-10 16:57:03

Fig 10.5 : Registrar interface for checking agents performance.

View Tutors

Tutor ID: Tutor Name: Search

Record New Tutor Details

#	Name	ID No	Duty	Qualification	Appointment date	Gender	Email	Phone No	Center
1	Mugo Joseph Gatheru	12345678	Lecturer	MSc	2009-09-09	Male	josephsps@g.com		KSPS
2	Muthe Gabriel Waibuu	129090	teach and evaluate	BSC IT JKUAT	0000-00-00	Male	rpwnjuguna@gmail.com		KSPS
3	NDAU Antony	1909	lecturing and student evaluation	bsc it	0000-00-00	Male	rpwnjuguna@gmail.com	+254722258483	ALFAX
4	KENNEDY KIARIE	1019	teaching	teaching evaluation and	0000-00-00	Male	rpwnjuguna@yahoo.com		KSPS
5	OSIEMO DIANA MORAA	129090	coordination	coordination	0000-00-00	Male	osiemo		KSPS
6	OWUOR SALOME ATIENO	109090	examination	MSC IS	0000-00-00	Female	owuor@ksp.ac.ke		ALFAX
7	Patrick Nyuguna	23792236			0000-00-00	Select	rpwnjuguna@gmail.com		ALFAX

15 Figure 10.6 List of Tutors

View Tutors

Tutor ID: Tutor Name: Search

#	Name	ID No	Duty	Qualification	Appointment date	Gender	Email	Phone No
1	Naivasha Patrick	107261	teaching and project evaluation	BSc math comp	2010-03-31	Male	nairasha@yahoo.com	+254721300644
2	Kyenge Bonifac	1010190		MSc IS	0000-00-00	Male	kyengeksp@gmail.com	+2542123850

Fig 10.7 List of tutor as accessed by center coordinator.

Center coordinator can allocate unit to teach in that center as shown in fig 10.8 below once allocated that information can be accessed. Coordinator agent use that data to alert concern tutor. Unit tutor are informed of allocation through text and email, they can also access the same through AB3S interface as shown in fig 10.9 below.

Unit tutors

Unit Code: Unit Name: Tutor Name: Search

Allocate Unit to Tutor

#	Unit	Semester	Year	Tutor	Num of Hours
1	BCT 2200 Project		0	KENNEDY KIARIE	5
2	CIT 11 Foundation maths		0	OSIEMO DIANA MORAA	5
3	BCT 2112 Accounting Principles		0	Nyaga joseph	5
4	BIT 2118 Application programming 1		0	Kyenge Boniface	5
5	CT14 Programming Concepts		0	Nyaga joseph	4
6	LD Land Dispute		0	KENNEDY KIARIE	5
7	CIL Company Law		0	Patrick Njuguna	0

16 Figure 10.8 Unit allocation to all tutors

Units Allocation

NAIVASHA PATRICK

#	Unit Code	Unit Name	No of Hours	of Appointment Date
1	BIT 2118	Application programming 1	5	2010-05-12
2	ICS 2104	Object oriented programming 1	5	2010-05-12

Fig 10.10 Units allocated to tutor.

Appendices B: Agent configuration interfaces

User-Agent interaction is important in AB3S , for specifying what agents needs to perform certain function. Such interfaces are as shown below.



Fig 10.11 Communications' agent interface

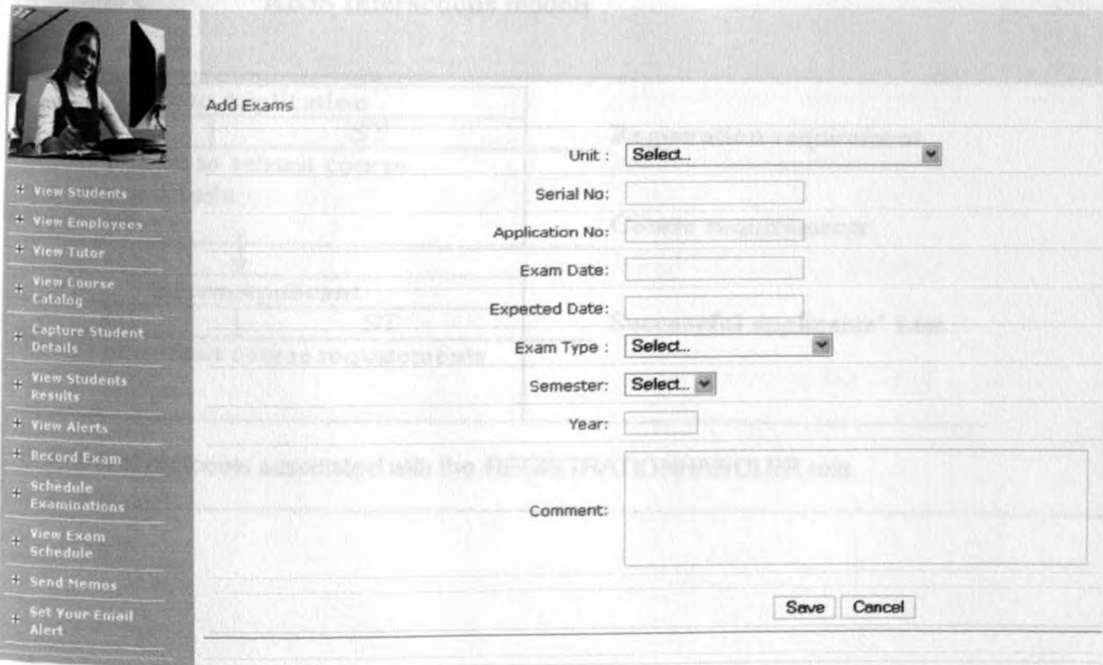


Fig 10.12 Coordinator agent interface.

From fig 10.12 once the agents have been configured the coordinator agent will be informing him/her of exam setting / submission progress as well as sending reminder to tutors based on that information.

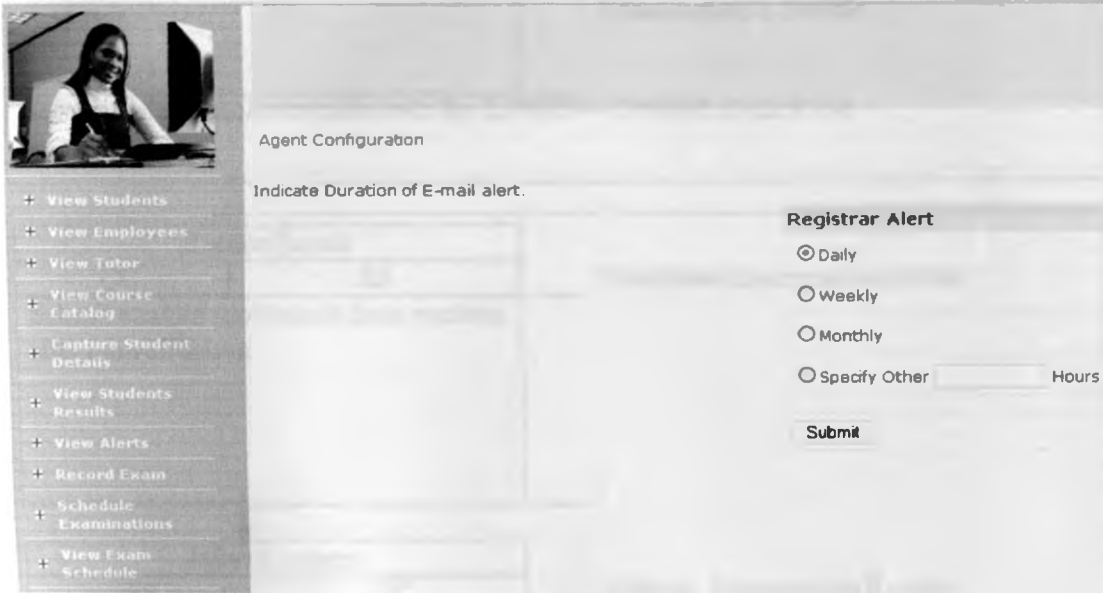
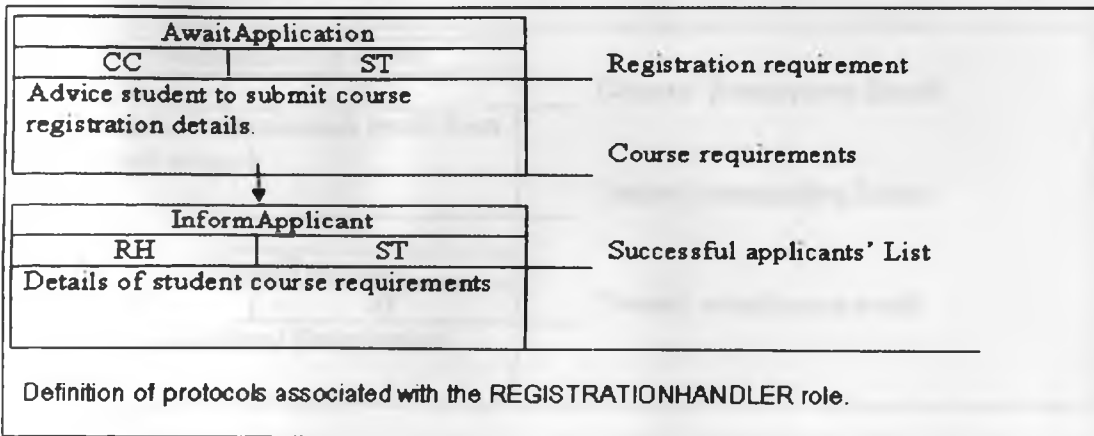


Fig 10.13. Agent configuration interface.

Appendices C AB3S Interactions models



InformCenters		
ES	CC	
Inform centers through course coordinator of examination schedules.		Examination schedule.
Definition of protocols associated with the EXAMINATIONSCHEDULER role.		

ReceiveResult		
CC	ES	Provisional Examination results
Submit examination result from centers after consolidation		
↓		
InformCenters		
ES	CC	Official Examination Results
Details of performance records sent to students through course coordinators		

ReceiveExaminationResult		
CT	CC	Courses' Examination Result
Receive course examination result from course instructors ↓		Centers' examination Result
InformResultProcessor		
CC	RP	Centers' examination result
Centers' consolidated Examination results		

InformCoordinator		Course Examination Result
CT	CC	
Submit students' course performance to coordinator for consolidation.		
Definition of protocols associated with the COURSEINSTRUCTOR role		

Appendices D AB3S Sample Codes

```
<?php
//sends a list of newly applied students to registrar and respective coordinators
//sends a list of accepted students to registrar and respective coordinators
//sends sms alerts and emails to accepted students
session_start();
require_once("config.php");
require_once("connection.php");
require_once("lib.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Students Agent</title>
</head>
<body>
<?php
$rs=retrieveNewlyAppliedStudentsByCenter();
$stud="";
$st="";
$js=0;
while($srow=mysql_fetch_object($rs))
{$students="";$i=0;
    $rs=retrieveCenterNewlyAppliedStudents($srow->centerid);
    while($srw=mysql_fetch_object($rs))
```

```

{Si++;Sj++;

    Sstudents=$i. ". $row->fname."&nbsp;"$row->onames."<br>";
    Sst=<li>"$row->fname."&nbsp;"$row->onames."</li>";
    mysql_query("update students set checked='Yes' where id='$row->id'");
}

//Sending list of newly applied students to every center
if($i>0)
{Scenter=retrieveCenter($row->centerid);

    $scood = retrieveCenterCoordinator($scenter->id);
    $semail=$scood->email;//$scenter->name."@ksps.ac.ke";
    $shaders="MIME-Version: 1.0\r\n";
    $shaders="Content-type: text/html; charset=iso-8859-1\r\n";
    ini_set("SMTP","10.2.28.5");

    ini_set("sendmail_from","rpwnjuguna@gmail.com");
    if(mail($semail,'Newly Applied Students to '.$scenter->name,$sstudents,$shaders))
    {
        recordAlert($scenter->id, $semail,"Email", "Newly applied
students","center","");
        echo"Email Sent to ".$semail;
    }
}

}

//sending an overall list of all applied students to registrar
if($j>0)
{$stud = "<ol>"$st."</ol>";

    $semail="rpwnjuguna@gmail.com";//$scenter->name."@ksps.ac.ke";

```



```

Sheaders="MIME-Version: 1.0\r\n";

Sheaders.="Content-type: text/html; charset=iso-8859-1\r\n";

ini_set("SMTP","10.2.28.5");

ini_set("sendmail_from","rpwnjuguna@gmail.com");

if(mail($email,'Overall Newly Applied Students sent to Registrar',$stud,$headers))
{
    recordAlert(0, $email,"Email","Newly registered students","registrar","");
    echo"Email Sent to ".$email;
}
}

$res=retrieveNewlyRegisteredStudentsByCenter();

$stud="";
$st="";
$j=0;
while($row=mysql_fetch_object($res))
{$students="";$i=0;
    $rs=retrieveCenterNewlyRegisteredStudents($row->centerid);
    while($rw=mysql_fetch_object($rs))
    {$i++;$j++;
        $course=retrieveCourse($rw->courseid);
        $students=$i." ".$rw->fname."&nbsp;".$rw->onames."&nbsp;".$rw->regno."&nbsp;".$course->code."&nbsp;".$course->name."<br>";
        $st="<i>".$rw->fname."&nbsp;".$rw->onames."&nbsp;".$rw->regno."&nbsp;<strong>".$course->code."</strong>&nbsp;".$course->name."</i>";
        mysql_query("update students set checked='Yes' where id='$rw->id'");
        //mysql_query("update students set acceptstatus='Yes' where id='$id'");
    }
}

```

```

//Sending list of newly registered students to every center
if($i>0)
{
    $center=retrieveCenter($row->centerid);

    $scood = retrieveCenterCoordinator($center->id);

    $semail=$scood->email;//$center->name."@ksps.ac.ke";

    $headers="MIME-Version: 1.0\r\n";

    $headers.="Content-type: text/html; charset=iso-8859-1\r\n";

    ini_set("SMTP","10.2.28.5");

    ini_set("sendmail_from","rpwnjuguna@gmail.com");

    if(mail($semail,'Newly Registered Students to '.$center->name,$students,$headers))
    {
        recordAlert($center->id, $semail,"Email","Newly registered
students","center","");

        echo"Email Sent to ".$semail;
    }
}
}

//sending an overall list of all registered students to registrar
if($j>0)
{$stud = "<ol>".$st."</ol>";

    $semail="rpwnjuguna@gmail.com";//$center->name."@ksps.ac.ke";

    $headers="MIME-Version: 1.0\r\n";

    $headers.="Content-type: text/html; charset=iso-8859-1\r\n";

    ini_set("SMTP","10.2.28.5");

    ini_set("sendmail_from","rpwnjuguna@gmail.com");

    if(mail($semail,'Overall Newly Registered Students sent to
Registrar',$stud,$headers))

```

```
{  
    recordAlert(0, $email,"Email","Newly registered students","registrar","");  
    echo"Email Sent to ".$email;  
}  
}  
  
//mysql_query("update students set checked='Yes' where id='$row->id'");  
  
?>  
</body>  
</html>
```