ONTOGRAPHY-BASED MULTIAGENT SYSTEM FOR PUBLIC SERVICE E-PROCUREMENT SHARED SERVICE

Denis W. Wahome
P58/75639/2012

Supervisor:
Mr. Eric Ayienga

A research project submitted to the School of Computing and Informatics in partial fulfilment of the requirement for the award of Masters of Science in Computer Science of the University of Nairobi.

November, 2014
Declaration

DECLARATION BY THE STUDENT

I, Denis Wandiaini Wahome, wish to confirm that this research project and the work presented in it is my original work and has not been presented for any other University award.

Denis Wandiaini Wahome
P58/75639/2012

DECLARATION BY THE SUPERVISORS

The Research Project has been submitted in partial fulfillment of the requirement of the Master of Science Degree in Computer Science of University of Nairobi with my approval as the University supervisor.

Mr. Eric Ayienga,
School of Computing and Informatics
University of Nairobi

DATE
Abstract

The research shows public service procurement is guided by public procurement and disposal act 2005 which stipulate the stages that government department must follow when procuring good or services. The stages are: pre-qualification, Identification of requirements, definition of requirements, determination of source, evaluation and selection of vendor, contract award, contract implementation/delivery, storage, payment and disposal. There has been attempt to automate the processes which of latest was to pre-qualifying the suppliers online starting August 2014. However, the process of tender evaluation has remained manual which is prone to human interference, tedious, time consuming, perusing volume of documents and difficult to adhere to 30% allocation of tenders to special group unless manual analysis is done. The research undertaken shows that ontology based multiagent system can be used in developing public procurement tender evaluation system. This was achievable because agents offer unique features of autonomy, social ability, reactivity, temporal continuity which lacks in object-oriented language. To achieve the goal of tender evaluation, a number of agents were identified to work together by sharing information seamlessly which was enabled by use of ontology’s embedded in jade environment. The agents were determined based on the processes undertaken before evaluation can be achieved. The processes included: identification of specification, generation of evaluation criteria, advertisement and response of tenders, technical and financial evaluation, and notification of bidders. Every stage was handled by agent which worked based on the knowledge it has while perceiving the environment. The prototype was evaluated using the business owner and response was positive. The results shows that, transparency, low cost, reduced tender evaluation time and allocation of 30% to youth, women and disabled in tender evaluation can be achieved through multiagent system. The study recommend piloting of the system in all government departments for them to appreciate and adopt use of software agents in tender awarding processes.
Acknowledgements

I wish to thank the Almighty God for standing with me and giving me the strength to pursue the studies to this end. I sincerely thank My wife Rosemary kangethe, my parents and my siblings for your great support. Without you I couldn’t have gone this far.

To my supervisor, Mr. Eric Ayienga, thank you for being available throughout my research project, listening, giving me support, advice and direction on my project.

To my project panelist Dr.Robert Oboko(panel Chairman), Prof.Okelo Odongo and Prof Omwenga,I salute you for your further contribution in my project.

I wish to thank Government of Kenya through Directorate of public service management for giving me opportunity to study procurement procedures and practices. I appreciate your key informants of my study data. To my colleagues Bernard Wekesa and Charles Mutinda, thank you for your support in various stages of the project.

I wish to appreciate others whom I have not mentioned here, who contributed to this project in one way or another.

May God bless you all.
# Table of Contents

**Declaration** .......................................................................................................................... i
**Abstract** ................................................................................................................................. ii
**Acknowledgements** ............................................................................................................... iii
**List of Tables** ........................................................................................................................... vii
**Abbreviations and Acronyms** ................................................................................................. viii

**CHAPTER ONE: INTRODUCTION ......................................................................................... 1**

1.0 Background .......................................................................................................................... 1
1.1 Problem Statement ............................................................................................................... 4
1.2 Objectives of the Study ......................................................................................................... 5
1.2.1 General Objectives .......................................................................................................... 5
1.2.2 Specific Objectives .......................................................................................................... 5
1.3 Research Questions .............................................................................................................. 5
1.4 Rationale of the Study .......................................................................................................... 5
1.5 Scope of the study .................................................................................................................. 6
1.6 Limitation of the Study ......................................................................................................... 6
1.7 Definitions of terms .............................................................................................................. 7

**TWO: LITRATURE REVIEW ................................................................................................. 8**

2.0 Overview .............................................................................................................................. 8
2.1 Agents and Multiagent systems .......................................................................................... 8
2.1.1 Agents ............................................................................................................................. 8
2.1.2 Multiagent System .......................................................................................................... 9
2.2.3 Ontology .......................................................................................................................... 11
2.2 Public procurement system in Kenya .................................................................................. 12
2.3 Related work ........................................................................................................................ 13
2.4 Summary .............................................................................................................................. 18

**CHAPTER THREE: METHODOLOGY ..................................................................................... 19**

3.0 Overview .............................................................................................................................. 19
3.1 Proposed Conceptual Framework ....................................................................................... 19
3.2 Multi-agent System Methodology ....................................................................................... 20
3.3 Prometheus Methodology ..................................................................................................... 20
3.4 System Agents ...................................................................................................................... 21
3.5 Agents conceptual model ..................................................................................................... 22
3.6 Development framework ..................................................................................................... 22
3.6 Requirements Specifications ................................................................................................. 22
3.7 Usability Evaluation ............................................................................................................ 24
List of figures

Figure 1: Ontology based MultiAgent System for business users (Lavbic, et al.2010) … 12
Figure 2: Framework for E-SCM Model using MAS (Al-zoubi, 2010) ………………..14
Figure 3: I-SEEC Architecture (Ezzeddine, et al.2012) ………………………………….17
Figure 4: Conceptual Framework …………………………………………………………19
Figure 5: PPOBMAS Agents conceptual model ………………………………………….22
Figure 6: Existing system Dataflow diagram ……………………………………………26
Figure 7: Proposed Use case diagram …………………………………………………….27
Figure 8: Agent system diagram ………………………………………………………….28
Figure 9: Agents Role Grouping ……………………………………………………......28
Figure 10: PPOBMAS Hierarchy Class diagram ……………………………………….30
Figure 11: PPOBMAS Entity Relation Diagram (ERD) …………………………………31
Figure 12: Launching of agent multiagents system ……………………………………34
Figure 13: Sniffer agents ……………………………………………………………….34
Figure 14: Web agent interact messages with specification agent …………………….36
Figure 15: Web agent interact with evaluation criteria agent …………………………36
Figure 16: Web agent interact with advertisement agent ………………………………37
Figure 17: Evaluation agent evaluating and Notifying bidders through Notification Agent37
Figure 18: Web Agent evaluation results ……………………………………………….38
List of Tables

Table 1: Based on Feldman and Yu (1999) ................................................................. 3
Table 2: Agents Responsibilities ................................................................................. 29
Table 3: Result table for system efficiency ................................................................. 38
Table 4: Result table for Affect ................................................................................ 39
Table 5: Result table for system Control .................................................................... 40
# Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Agent Communication Language</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligent</td>
</tr>
<tr>
<td>AMS</td>
<td>Agent Management System</td>
</tr>
<tr>
<td>Df</td>
<td>Directory Facilitator</td>
</tr>
<tr>
<td>ERD</td>
<td>Entity Relationship Diagram</td>
</tr>
<tr>
<td>FIPA</td>
<td>Foundation for Intelligent Physical Agents</td>
</tr>
<tr>
<td>GCCN</td>
<td>Government Core Common Network</td>
</tr>
<tr>
<td>GDP</td>
<td>Gross Domestic Product</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communications Technology</td>
</tr>
<tr>
<td>IFMIS</td>
<td>Integrated Financial Management Information System</td>
</tr>
<tr>
<td>JADE</td>
<td>Java Agent Development Environment</td>
</tr>
<tr>
<td>KQML</td>
<td>Knowledge Query and Manipulation Format</td>
</tr>
<tr>
<td>MAS</td>
<td>Multiagent System</td>
</tr>
<tr>
<td>PPOA</td>
<td>Public procurement Oversight Authority</td>
</tr>
<tr>
<td>PPOBMAS</td>
<td>Public procurement Ontology Based Multiagent System</td>
</tr>
<tr>
<td>RMA</td>
<td>Remote Management Agent</td>
</tr>
<tr>
<td>SUMI</td>
<td>Software Usability Measurement Inventory</td>
</tr>
</tbody>
</table>
CHAPTER ONE: INTRODUCTION

1.0 Background

The management of public resources demands a sound procurement procedures in order for its citizen to realize the value of their money. The Kenya government procurement system have undergone several evolution where the recent one was establishment of the public procurement and disposal Act 2005. Before the Act 2005, the procurement was governed by Treasury circulars from 1969, then the supplies manual of 1978 which was succeeded by promulgation of the Exchequer and Audit Regulations 2001. All this initiatives were aimed to improve the public procurement system by ensuring accountability and transparency in management of public resources.

Juma(2010) in his study stated the total value of public procurement in the central government is currently estimated at 10% of the Gross Domestic Product (GDP), accordingly, Kenya’s GDP in the year 2008 was estimated at Ksh. 2,099.79 billion putting the total expenditure on procurement by the Government at around Ksh. 209.79 billion annually. Even at 10% savings due to improvements in procurement practices and controls would mean a yearly gain to the exchequer of about Ksh. 21 Billions.

In reference to public procurement and disposal Act 2005, procurement process entails Identification of requirements, procurement planning, definition of requirements, determination of source, evaluation and selection of vendor, contract award, contract implementation/delivery, storage, payment and disposal. This shows procurement information system should be developed based on the above requirement. Corruption prevention guidelines in public procurement(2009) states that there is manifestation of corruption in all above stages of procurement process which can be addressed with sound technological system and integrity of supply chain officers.

In February 2012, His Excellency the Retired President, Hon. Mwai Kibaki directed that 10% of all Government contracts be earmarked and awarded to the youth. The Policy directive was informed by the Government’s realization that in order to meaningfully address the issue of youth unemployment, it is necessary to give them opportunities to participate in government contracts and tenders. In 2013, His Excellency the President Uhuru Kenyatta pledged that the procurement rules would be amended to allow 30 per cent of contracts to be given to the youth, women and persons with disability without
competition from established firms and also directed that the issue of 30 per cent allocation of all Government procurement to the youth should be adhered to, warning those who will fail to effect the directive will be sacked. Procurement system therefore, continue to remain key economic player in realisation of vision 2030.

With the introduction of IFMIS by Ministry of finance in year 2003, one of objective was to incorporate procurement module within the system from the first stage of procurement process to the last point of payment. The current system is intranet based accessed centrally by all ministry departments in a common core network. However the system can only capture commitment of money for works/service and initiate invoice payment thus leaving the process of prequalification, requirement definition, evaluation criteria, tenders and quotation advertisement, technical and financial evaluation, communication of highest bidder and procurement planning manual. In spite of the system being centralised, no sharing of information across the departments which can assist them to make wise decision when procuring specialised goods/services. In order to realise 30% awarding of tenders to youth, women and disabled, there must be a control measures in the system to ensure no group surpasses its percentage limit.

Ontology based multiagent system is a powerful technology where agents interact by exchanging information without much involvement of human being. Ontology ensure seamless sharing of information among the agents. This body of knowledge is envisaged to provide a more effective and efficient tender evaluation shared services across Government departments where minimal human interaction will be required especially on ensuring transparency and accountability (Wooldridge, 2002).

The multiagent systems are distinguished by offering the following benefits as compared to other software programs:

- autonomy: agent works without interference of human being and it control its action and the internal state. The agents in procurement system will be expected to work independent;
- social ability: agents interacts with other agents and humans through agent communication language (ACL).ontology will ensure seamless sharing of information among the agents because no agent have full control of the system;
- reactivity: agents perceive their environment which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined, and respond in a timely manner to changes that occur in it;

- temporal continuity: Unlike much conventional software, which performs a fixed task then terminates, agents run constantly. This allows agents to monitor the current situation and take appropriate action when required. (Herman, 1997):

### EXHIBIT D.1 Software Agents vs. Traditional Software Programs

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Regular Software</th>
<th>Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature</td>
<td>Static</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Manipulation</td>
<td>Manipulation</td>
<td>Indirect: Autonomous</td>
</tr>
<tr>
<td>Interactivity</td>
<td>Noninteractive</td>
<td>Dialogues are fully interactive. Actions may be initiated by either the user or the agent system..</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Never changes, unless changed by a human or an error in the program</td>
<td>Adapts, learns</td>
</tr>
<tr>
<td>Temporal continuity</td>
<td>Runs one time, then stops to be run again when called.</td>
<td>Persistent: Continues to run over time.</td>
</tr>
<tr>
<td>Response</td>
<td>Predictable: Does what you tell it, what you said.</td>
<td>Interprets what you mean. Actions are based on rules but they may change over time.</td>
</tr>
<tr>
<td>Independence</td>
<td>Instructions</td>
<td>May initiate actions, as well as respond to instructions</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Generates process in one processing power</td>
<td>Dispatches simultaneously to accomplish various parts dedicated server with limited of a task in parallel</td>
</tr>
<tr>
<td>Local interaction</td>
<td>Accesses data across network using client-server architecture</td>
<td>Can travel and interact with local entities, such as databases, file servers and stationary agent, through message passing</td>
</tr>
</tbody>
</table>

Table 1: Based on Feldman and Yu (1999).
The rest of chapter will discuss the problem statement, rationale of the study, objectives of the study, research question, scope of the study assumption and limitation of the study and last but not least definitions of important terms.

1.1 Problem Statement
The current government administration which came on platform of doing business differently, the call for efficient and effective management of procurement has been overemphasized. In addition, Section 227 of the Constitution of Kenya provides for establishment of a system for procurement of goods and services that is fair, equitable, transparent, competitive and cost-effective. The expectation of the government is to ensure youth, women and disability are allocated 30% of the tender in public institution and enhance transparency in awarding of contracts. Further, sharing of information on how and where to procure specialized goods/services should be done across Ministry/Department/Agencies.

Currently, the existing procurement system being used in the Government departments doesn't address above expectations. The system at the moment can only capture the commitment of the funds to procure good and services and downloading of tenders documents using ifmis portal leaving other process manual. The evaluation of tenders is done by committee composed of officers drawn from different department which some don't have knowledge on requirement specification of the good or service being procured. This compromise the objectivity of tender evaluation exercise. Further, It's difficult to determine the allocation of tenders to youth, women and person with disability as year progress until manual analysis is done. The pricing of similar good and services is done differently across the departments and there is no sharing of information across departments which can assist in making well informed decision when engaging in new contracts for the first time.

The research therefore, aim to demonstrate how ontology based multiagent systems can be used to address transparency in tender evaluation process, ensure allocation of 30% of tenders to youth, women and person with disability, and show how to shorten period of tender evaluation and reduce cost. This will be achieved by use of unique features of agents i.e. autonomy, social ability, reactive and temporal continuity. This features lack in information systems thus the choice of multiagents system.
1.2 Objectives of the Study

1.2.1 General Objectives

To provide an ontology based multiagent systems for public procurement tender evaluation process.

1.2.2 Specific Objectives

1. To research on how the features and environments of ontology, agent and multi agents system can be used in public e-procurement shared service.
2. To develop a multiagent system framework for e-procurement tender evaluation process.
3. To design and develop an ontology-based multiagent system prototype.
4. To evaluate the usability and practicality of system prototype using Ministry staff.

1.3 Research Questions

The proposed study will be guided by the following key research questions.
1. What are key features and environment of ontology, agents and multi-agents system?
2. How to develop a multiagent system framework for e-procurement tender evaluation process?
3. How to design and develop an ontology-based multiagent system prototype?
4. How to evaluate the usability and practicality of system prototype using Ministry staff?

1.4 Rationale of the Study

The government have come up with several legal regulations, directives from the President touching on procurement procedures framework. However, without a proper technological implementation strategies, the goal of regulations and directive will not be achieved as expected. The study aims to assist government department meets its obligation of efficient management of procurement system by providing controls to ensure 30% allocation of contracts to youth, women and disabled, reduce human interaction in procurement process, participation of suppliers in the whole process of awarding contract through online interaction thus ensuring transparency and accountability, sharing of information across the departments on better ways to initiate a new contract for specialized goods/services, proper projection of procurement planning which are aligned with budget allocation. The public procurement oversight authority(PPOA) will be able to analysis easily adherence to
procurement procedures by aggregating information across government departments and finally reduce turnaround time of delivery of procurement services.

The study will examine how ontology-based multi agent system can be used in a procurement shared services. Further the research findings, will help other researchers who would want to undertake the same study to come up with more comprehensive research work or highlight the necessary adjustment that will not have been dealt with in depth by this research.

1.5 Scope of the study
The research will be limited to tender evaluation process in the public service departments. This will entail requirement definition, tender/Quotation advertisement, application for tender/quotation, evaluation criteria, evaluating the bidders by considering 30% allocation to special groups and notifying the bidders their result. Due to time constraints for creating and testing an inference engine of mega projects in the Government, the prototype will demonstrate the use of agents in awarding recurrent expenditure contracts. However, the prototype can be used both in recurrent and development expenditure contracts.

1.6 Limitation of the Study
Although consideration will be made to ensure that the data will be valid and findings are reliable, nevertheless there could be some errors. These include:

I. the study will rely on secondary data for information processing, the presence of faulty data definitely might pose a problem to the study;

II. the design and the execution of this research work lies on the scarcity of resources in the form of financial constraints, time constraint and materials needed for the project work;

III. some respondents might view the study as more of audit than an academic study and thought findings of the study would be going to the higher authorities who would know their short falls and influence them in the way and manner they will respond to some of the questions; and

IV. Furthermore, this research does not claim comprehensiveness, therefore there will be areas suggested for further research to fill whatever gaps this project will leave.
1.7 Definitions of terms

**Procurement**: means the acquisition by purchase, rental, lease, hire purchase, license, tenancy, franchise, or by any other contractual means of any type of works, assets, services or goods including livestock or any combination.

**E-Procurement**: The use of Web-based technology to support the key procurement processes, including requisitioning, sourcing, contracting, ordering, and payment.

**Supply chain management**: is the management of the flow of goods. It includes the movement and storage of raw materials, work-in-process inventory, and finished goods from point of origin to point of consumption.

**JADE** – This is Java Agent Development Framework, is a software framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with FIPA specifications and through a set of graphical tools that support the debugging and deployment phases.

**FIPA**: Foundation for Intelligent Physical Agents. The FIPA 97 specification is the first output of the Foundation for Intelligent Physical Agents. It provides specification of basic agent technologies that can be integrated by agent systems developers to make complex systems with a high degree of interoperability.

**Software agent**: Are components of software or hardware, which are capable of acting exactly in order to accomplish tasks on behalf of their users.

**Quotation**: is a procedure for seeking a price and delivery time for goods which are easily specified.

**Tenders**: its formal and detailed exercise of obtaining sealed bids for goods where the value is high and/or the specification is detailed or complex and/or special terms and conditions may apply.

**Ontology**: are content theories about objects their properties and relationships among them that are possible in a specific domain of knowledge. In multiagents, ontology ensure seamless communication between agents.

**Multiagent system**: It's a collection of autonomous agents that can define their own goals and actions and can interact and collaborate among each other through communication.

**Shared service**: is an accountable entity within a multi-unit organization tasked with supplying the business unit, respective divisions and departments with specialized services.
TWO: LITERATURE REVIEW

2.0 Overview
The chapter begins with the discussion of agents, multi agent system, ontology in multiagent system and previous related work on use of ontology-based multi agent systems in procurement. The chapter ends by summarizing the contribution of ontology based multiagent system e-procurement shared services in Government departments.

2.1 Agents and Multiagent systems
Multiagent systems emanated as a scientific domain from previous researches in Distributed Artificial Intelligence which started in the early 80's. Further, MAS is now seen as a major trend in research and development mainly related to artificial intelligence and distributed computing techniques (Oliveira, et al. 1999).

2.1.1 Agents
Among the researchers there have been no universally accepted definition of agents. However, recent research (Jennings, et al. 1996) defines an agent "is a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives ". In his study on intelligent software agent in the internet, Herman (1997) state that an agent is a "piece of software which performs a given task using information gleaned from its environment to act in a suitable manner so as to complete the task successfully. The software should be able to adapt itself based on changes occurring in its environment, so that a change in circumstances will still yield the intended result ". Herman further argue that the agent must possess the following characteristics:

- autonomy: agent should work without interference of human being and it control its action and the internal state;
- social ability: agents should interact with other agents and humans through agent communication language (ACL);
- reactivity: agents perceive their environment which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined, and respond in a timely manner to changes that occur in it;
- proactively: agents exhibit goal-directed behavior by taking the initiative within the environment they work on;
temporal continuity: Unlike much conventional software, which performs a fixed task then terminates, agents run constantly. This allows agents to monitor the current situation and take appropriate action when required.;

goal-oriented: an agent is capable of handling complex, high-level tasks.

2.1.2 Multiagent System

Multiagent system is a group of agents where single agent does not possess enough knowledge to solve the problem on its own; no common management system; data is decentralized; and computing is executed in parallel way (Wooldridge, 2002).

Agent communication: In a multi-Agent system the knowledge and function of the system is distributed among the agents. Agents must have methods for sharing knowledge and taking advantage of each other's capabilities. It would not be feasible for every agent to have complete knowledge of the entire system, nor to have all possible capabilities. This necessitates the need of communication language. In their work (Genesereth and ketchpel, 1994) state that Agent communication language consist of three parts: vocabulary, an internal language called knowledge Interchange Format(KIF) and external language called knowledge Query and Manipulation Format(KQML). Agent Communication Language(ACL) message is a KQML expression in which the arguments are terms or sentences in KIF formed from words in the ACL vocabulary.

Agent coordination: In their work on The Interdisciplinary Study of Coordination (Malone and Crowson, 1994) defined coordination as the act of managing interdependencies between activities, and classified dependencies into four categories: shared resource, producer/consumer relationship, simultaneity constraint and task/subtask. They are factors that require the multi agents system to coordinate. In their study (Nwana, et el, 1996) gave the following factors:

- Avoiding chaos: With the decentralisation in agent-based systems, the agents are prone to chaos or anarchy;
- Efficiency: Even if agents can work independently, sometimes working together means being able to solve problems faster;
- Meeting global constraints: If there are budget limits for perform task, then agents must make agreements so that they do not unintentionally exceed the budget limit provided;
• Distributed information, expertise or resources: An agent doesn't possess all the knowledge or information for performing task, thus there is need to coordinate the agent in performing collective task by sharing information or resources; and
• Dependencies between the agents’ actions: This occur when an agent to perform its task, another agent must complete its task first. This call for coordinating the agents in order for the to achieve their goal seamlessly.

Agent classification: Researchers have come up with a number of agent classifications where many of them have either focused on a particular domain or on a specific type of agent. In the study A New Classification Scheme for Software Agents (Hector and Narasimhan, 2005) proposed new classification which they borrowed much from previous researches of: Wooldridge and Jennings (1995); Franklin and Graesser (1996); and Parunak (1996).

The classification are:
• Proactive-This is how the agents reacts to and reasons about its surrounding, and how it works towards its goals. This agents can be considered either pure reactive, pure planning or hybrid.
• Adaptive. It's the ability of an agent to changes its behavior over period time. However, its closely related to pro-activeness, with many pure planning or hybrid systems relying on the ability to adapt. Despite this, not all agents are adaptive, and some only adapt in a limited manner. Adaptiveness may fall into several different categories either as learning, subsumption, Non-adaptive or constrained based.
• Mobility: Software agents have a capability to carry out tasks involving large-distributed networks such as the Internet. Many researchers have exploited the mobility of an agent in their work. An agent may be either physically mobile, logically mobile or static.
• Collaboration: Collaboration between agents underscores the success of an operation or action in an expected time. To achieve this, the agents should have some form of social ability, and this may be divided into communicative and non communicative.
• Disposition: In this classification, agents are grouped according to attitude of the agent toward other agents, and its willingness to cooperate with other agents. Agents may be classified as benevolent, self-interested or malevolent.
Veracity: Where agents are collaborating, either communicative or non-communicative, may try to deceive other agents via their messages or behavior. Agents may therefore be classified by their veracity i.e. truthfulness or untruthfulness.

2.2.3 Ontology
Ontology is a content theory about objects, their properties and relationships among them that are possible in a specific field of knowledge. In knowledge-based systems, an ontology is that part of the system which specifies what things exist and what is true about them. Ontology's enable knowledge sharing since they capture the essential conceptual structure of the field using a suitable knowledge representation language. The language can be shared among others that have similar needs for knowledge representation in that field thus avoiding the need for replicating the knowledge analysis process (Chandrasekaran et al., 1999).

The use of ontology's in Multi-Agent System (MAS) environment enables agents to share a common set of concepts about framework i.e. user profiles, products and other domain elements while interacting with each other. Agents have a capability of exploiting the existing reasoning mechanisms to infer derived contexts from known background, to make decisions and to adapt to the surrounding environment, current status, and personal setting of the user. In their example on a DSS-MAS for mobile telecommunication, they defined several sub task and later described by ontology as shown below in DSS-MAS architecture (Lavbić et al., 2010).
2.2 Public procurement system in Kenya

E-procurement is a phenomenon that started in the developed economies but due to the growth of the internet, it has been increasingly adopted in the developing countries. Organizations and individual business people have come to embrace it due to the integration benefits and the immense possibilities it brings about: removing space, distance and time (European Union, 2012). Procurement has become popular to implement in the latest era by both governments and enterprises. Although the opportunities for improvement seem abound, both private and public sector are still cautious as far as the adoption of electronic technologies is concerned (Zheng, et al., 2004).
Public procurement system in Kenya has undergone significant developments. From being a system with no regulations in the 1960s, and a system regulated by Treasury Circulars in the 1970s, 1980s and 1990s. The introduction of the Public Procurement and Disposal Act (PPDA) of 2005 and the Procurement Regulations of 2006 has introduced new standards for public procurement in Kenya (PPOA, 2007).

Procurement process entails Identification of requirements, procurement planning, definition of requirements, determination of source, evaluation and selection of vendor, contract award, contract implementation/delivery, storage, payment and disposal (Public Procurement and Disposal Act 2005). Corruption prevention guidelines in public procurement (2009) states that there is manifestation of corruption in all above stages of procurement process.

Section 227 of the Constitution of Kenya provides for establishment of a system for procurement of goods and services that is fair, equitable, transparent, competitive and cost-effective.

For any e-procurement initiative to be successful, there are a number of factors that an organization must critically consider. They include: user acceptance of new information system; information quality; trust; risk perception; early supplier involvement; staff training; users and buyers; compliance with best practices; top management support; continuous measurement of the key benefits; re-designing affected business processes and actual selection of e-procurement solution.

2.3 Related work
MULEP-a which is a Multi Level E-Procurement framework where the procurement process is fulfilled via distributed mobile agents developed over Java Agent Development Framework-JADE. The MULEP-a work in an environment where there is several buyers and suppliers who spread all over the world and interacting via the Internet. There is a main controller running at central host and several remote containers running on remote host. The controller has their own execution flow and stationary agents. In the framework the mobile agents are used as mediator between buyer and the seller. The MULEP-a have the following flow: There are buyers and suppliers registered in the system. For the buyer to buy a product, he has to make a request from main controller through online interface. The stationary agents generates a mobile agent which searches the best price and return the result to main container. When a supplier receive a mobile agent request, the agent get activated and communicates with the stationary supplier agent to decide the next action.
At this point the mobile gets either a price from the supplier agent and returns result to one up level or Supplier agent returns it does not have the requested products right now, but they can be obtained from one sub-level supplier. When all results are collected from the sub level suppliers, mobile agent selects the best dealing one and sends this message to up level mobile agent. The final stage the agent at the main container gets the all necessary prices and select best price and then sends the approval message to suppliers directly or through multi-levelled mobile agents (Sahingoz and Oztas, 2009).

The gap in the framework has not been applied in different application domains and there is need to focus on new design decisions to improve its performance and scalability because of its internet working environment.

Electronic supply chain Management using multiagent system called MAS-SCM consists of several different functional agents that perform functions of the supply chain management. The agents have common understanding of system ontology and uses KQML to make conversation. The model uses ontology to define the knowledge of goods that the system is dealing with and the interaction rules. The framework has predefined number of information agents which are in charge of providing different system information where a major task of an agent is to assign a value to each variable that satisfies all the control according to their own rules or orders from the customers. The model has the following functional agents: Manager Agent (MA) Purchasing Agent (PA), Manufacturing Agent (FA), Selling Agent (SA) and Inventory Agent (VA).

Figure 2: Framework for E-SCM Model using MAS (Al-zoubi, 2010). Selling Agent associates and interacts with outside purchasing agents owned by distributors, retailers and customers, which contains the controls for supply and maintaining quote. The selling agent is also responsible of receiving orders and delivering
goods to customers. Purchasing Agent interacts with outside selling agents owned by suppliers, which contains the controls for order management, filling orders and deliver materials to the manufacturer. Manufacturing Agent controls the manufacturing process, which contains the controls for monitoring the operation, production scheduling, and monitoring the quantity of raw materials. Inventory Agent (VA) controls the inventory levels, which contains the constraint for monitoring inventory flow, acquire information from manager agent to calculate needed materials based on historical data for optimal reorder quantities, also determines inventory replenishment policy; generate delivery plan, and safety stock. Manager Agent (MA) manages and controls all the agents by using local Database which stores all related information (Al-zoubi, 2010).

The study shows that most Electronic Supply Chain Management are still in the early stage of theoretical framework or demonstrating prototypes. The MAS-SCM framework has a number of gaps which include: how to determine the tasks of each functional agent; developing a standard ontology for the multi-agent system; and how to develop a standard agent-based language that could be applied within common internet based protocols.

Uppin and Hebbal (2010) described a multi agent system model for designing supply chain which ensure efficient sharing of information in an integrated functioning of various units of an organization. The model has a number agent performing independent role. In the model, stock agent is used for storing information about the available stock and to give access to the data to other agents through negotiation or sending standard massages. The agent also updates the data before and after each purchasing activity. Master production schedule agent is used to receive the data and information about customer orders, forecast, available capacity from the concerned agents and transfer its output to time phasing agent. Agent for rough cut capacity planning provides a rough cut capacity plan which will be developed after collecting the relevant data about the recourses from the related agents. The agent also prepares the capacity planning information required for agents of production planning activities in particular scheduling. Product data agent is responsible for product data acquisition from product design and development agent regarding product structure, dimensions, tolerances and needed specifications. Agent for bill of material gets the required information about a given part from product data agent for processing and computing the bill of material of the part. Agent for time phasing collect information for its tasks from other agents related to bills of material, product data, stores, manufacturing production schedules. It processes the information and through time
phasing of planned orders, it determine two lists i.e. period wise list of items to be purchased and period wise list of items to be manufactured. Vendor data Agent is independent agent interacting with the different suppliers available in the network and acquires information about the suppliers for future considerations. The information includes cost, delivery, quality and purchase terms. It presents the data as item wise list of potential sellers.

In the work, part of the model related to purchasing activities is highlighted with suitable results. However other parts of the model such as functions related to process planning and scheduling activities for the list of items to be processed are not presented. The study concentrated on showing how information sharing among agents is important for effective supply chain management.

Srinivasan, et al (2011) demonstrated a Multi-agent System based Service Oriented Architecture for Supply Chain Management System (MAS-SOA-SCM). The multiagent system provides a real time application while service oriented architecture provides solution for manufacturers in turning the process of managing inventory policies into a competitive weapon, to improve service, reduce costs and gain market share. MAS-SOA-SCM works in manufacturer supply chain environment where it receives orders from customers and interacts with manager agent through order processing agent. Production agent fulfils the order through scheduling agent and delivery agent. Where more production is required, scheduling agent coordinates with inventory agent. Inventory agent takes care of the supply of raw materials to production or coordinate with supplier agent for further negotiation with suppliers. MAS-SOA-SCM desires to meet the needs of the customers on time and eliminate the need for a large inventory at the manufacturer and thus reduce overall cost in the entire manufacturing system.

In spite of MAS-SOA-SCM combining Multi-agent system and service oriented architecture to provide a supply chain management system which improve the efficiency, reduce the cost by controlling the inventory, reducing the impact of demand-supply problems and setting up proper negotiation policy, the model still have some gaps on security mechanism and the design approach, BPEL language is used in the business logic part of the system where it's still evolving and Service oriented architecture developer’s tools are not mature enough for solving some integration problems between user interface and business logic. In addition the framework didn't demonstrate competitive process of getting raw materials from suppliers.
Ezzeddine, et el.(2012) proposed a agent cooperation framework for supply chain and called it Intelligent System for Extended Enterprise Cooperation (I-SEEC). The I-SEEC uses multiagent system and semantic web services for collaborating decision making in the entire organization. The model has several agents playing a different role. The administrator agent play a role of managing user accounts that wishes to be added into the system in order to achieve a common goal. The customer agent provide an interface for entering customer request in the system and displaying results of a request. When customer agent discover the request, it forward to discovery agent for action. The provider agent provides an interface between I-SEEC and the provider of the Web Service. Service agent is used to intervene the time of negotiation with the customer in order to come up with a contract. The discovery agent looks for semantic services web appropriate to the customer agent requirements. The selection agent research on the best service to satisfy the customer requirements.

![I-SEEC Architecture](https://example.com/i-seec-architecture.png)

**Figure 3:** I-SEEC Architecture(Ezzeddine, et el.2012)

The broker agent deals with the balance of the work and the monitoring of the execution process of the service. It is also responsible for managing interoperability. The model has a number of warehouses for ontology which are: domain ontology for conceptualization of the specific field; application ontology which contains concepts depending on a field and a particular task; negotiation ontology which provides concepts and the rules used for the description of the protocol for negotiating contract; and local knowledge ontology which
contains knowledge about environment of each agent. There two type of Semantic Web Service register i.e. central and local. The central contains the Semantic description of the various services for the actors of the Supply Chain distribution while local represents a copy of the central register of services of the providers which allow storage of the services requested by the customers and the providers.

The architecture did not address supply chain of an organisation guided by procurement rules where tendering process must be followed to award a contract. In addition the study did not develop a prototype to validate the performance of the proposed model.

2.4 Summary

The study shows that multiagent system play a critical role in efficiency of supply chain management. The use of multiagent system in supply chain have been attributed to the ability of agent being autonomy, having social mobility and flexibility which is the key to supply chain processes. The study reveal that ontology can be used for information sharing among the agents in a distributed environment. Most of researchers concentrated on use of multiagent in a production environment where inefficiency in supply chain will severely affect the performance of an organization.

Further, the previous study shows that agents were limited to direct purchasing from suppliers without following any procurement procedures of tendering and awarding of contract which is critical to public service oriented organization.

In the proposed study, we want to develop a ontology based multiagent framework which will meet the requirement of dynamic public service oriented organization as opposed to limitation of previous frameworks which are not oriented to Government procurement procedures. As contribution, the framework will show how ontology based multiagent can be used to adherence to 30% awarding of contract to youth, women & disabled people; sharing of information on how to procure specialised goods/services across Government departments; ensure minimal human interaction in tender evaluation. The study will also add knowledge on use of multiagent system in a legal guided procurement process in public service oriented organisation.
CHAPTER THREE: METHODOLOGY

3.0 Overview
This chapter discusses the methodology that was used to achieve the research objectives of the study.

3.1 Proposed Conceptual Framework
The proposed ontology based multiagent system for public service e-procurement shared service will adopt the following conceptual framework.

![Conceptual Framework Diagram]

Figure 4: Conceptual Framework
3.2 Multi-agent System Methodology
Nowadays, agent-based technologies are considered to be the most promising means to deploy enterprise wide and world wide applications that often must operate across corporations and continents and inter-operate with other heterogeneous systems. A MAS Methodology generally encompasses a set of concepts, notations for modelling aspects of the software and processes that follow in order to build the software. The study adopted an agent oriented methodology called Prometheus.

3.2.1 Agent Oriented Methodology
An agent is a software concept. The growth in need of agent-oriented solutions in several areas of software engineering resulted in development of agent oriented methodology. There are numerous methodologies for agent-oriented. Some are based on ideas from artificial intelligence (AI), others as direct extensions of existing OO methodologies. Features of agent methodologies are as follows

- Provide sufficient abstractions to fully model and support agents and Multiagent systems.
- focus on an organized society of agents playing roles within an environment
- Support MAS, where agents interact according to protocols determined by the agents’ roles
- be agent-oriented in that it is geared towards the creation of agent-based software

3.3 Prometheus Methodology
Prometheus methodology was used in the system specification, system design and implementation. It is a detailed process for specifying, designing, and implementing intelligent agent systems. This methodology distinguishes itself from other methodologies by supporting the development of intelligent agents, providing start-to-end support, having evolved out of practical industrial, having been used in both industry and academia, and, above all, being detailed and complete. Prometheus is also amenable to tool support and provides scope for cross checking between designs. Prometheus methodology provides everything that is needed for defining and designing agents and It also covers a range of activities including the determining requirements until accurate design step. It is composed of three phases.

- **System Specification**: where the system is specified using goals and use case scenarios; the system’s interface to its environment is described in terms of actions, percepts, and external data; and functionalities are defined.
• **Architectural design:** where agent types are identified; the system’s overall structure is captured in a system overview diagram; and use case scenarios are developed into interaction protocols.

• **Detailed design:** 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.

### 3.4 System Agents

The proposed ontology-based multi-agent system adopted an e-procurement shared services framework. The framework have the following agents:

- **Specification Agent** it defines and provides the requirement for engaging the consultant/suppliers i.e. specification and term of references.

- **Advertising Agent** It publishes the tender to the public for response and automatically closes the tender when tendering period expires.

- **Evaluation Criteria Agent** the agent defines and provides the criteria to be used when evaluating the tenders.

- **Evaluation Agent** the agent carry out of evaluation of respondent on a given tender. It consider 30% allocation to special group.

- **Notification Agent** It communicate the evaluation results to the bidders.

- **Web agent**: It act an interface between users and other agents in the system.

- **RMA, DF and AMS agents**, these are system agents, shipped with the JADE distribution. The **AMS** (Agent Management System) agent is the authority in the platform. It is the only agent that can create and kill other agents, kill containers, and shut down the platform. The **DF** (Directory Facilitator) agent implements a yellow pages service which advertises the services of agents in the platform so other agents requiring those services can find them.

- **Persistence Agent**, which handles database operations for the platform.
3.5 Agents conceptual model

![Diagram of PPOBMAS Agents conceptual model]

3.6 Development framework
The multiagent system will be implemented using Jade. JADE (Java Agent Development Framework) is a software Framework fully implemented in Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. The agent platform can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. Net beans an integrated development environment will be used to code java classes then compiled in jade.

3.6 Requirements Specifications
Requirements specification involved designing of activities that would give the overall goals and more specific requirements for design of public procurement tender evaluation
ontology based multiagent system. The main goal of the prototype was to show how tender evaluation can be done by agents without involving human interaction in determining the bid winners. It involved the following:

a. Requirements Gathering
A target group was identified which included supply chain officers who are in charge of procuring of good and services, administrators who ensure staff are provided with tools of doing business and ICT officers who facilitate the use of ICT in service delivery.

The requirements were identified which included accessing the operations of procurement entity, discussing with supply chain on challenges they face in using existing system, ICT officers provided an insert on level of using ICT in organisation and challenges they are facing. The administrator was interviewed to assess the cost implication of running existing system and attempts which has been made to improve the processes.

We realized that the means that the existing systems provide tender evaluation was not efficient in service delivery because of manual processes involved. This led to identification of our agents application requirements which included: an application that can use agents in tender evaluation considering 30% allocation to youth, women and disabled without using human interaction.

b. Application data: - Research Methodology
The research used purposive interview approach to get content for the system prototype application. The content was gathered from the key informants using question guided interview (Sample questions attached in the appendix). The desired variables were identified and their viability established to form basis of the agents interaction and the user requirements. The prototype was developed with the necessary information on efficient way of doing tender evaluation starting from requirement identification to notification of bidders results.

c. Data collection
This involved collecting relevant information for procurement procedures as stipulated in public procurement and disposal Act 2005. This information was collected from supply chain officers in the Directorate of public service management (DPSM). Data on specifications and evaluation criteria was collected from ICT manager. The ICT department was chosen because it procure goods and services to be used by all departments and because of time constraints for evaluating the prototype. The data was collected using interview guided questions and also observation.
d. Data Analysis
The data collected was analyzed so as to get information for the prototype which would be useful in achieving the goal of agents doing tender evaluation. Analysis was also done for the application content, user’s characteristics, determination of 30% allocation to youth, women and disable persons and goal analysis.

The qualitative data collected was analyzed based on the following stages:

i. Requirement definition
ii. Evaluation criteria
iii. Making tender advert.
iv. Evaluation and;
v. Notification of bidders.

The above stages formed the basis of creating agents which working together achieved the objectives of the research study.

3.7 Usability Evaluation
Usability evaluation was conducted to assess the implementation of the prototype. This was a user based evaluation that measured perceived usability by users. This was guided by Software Usability Measurement Inventory (SUMI) questionnaire which is an internationally-standardized questionnaire for quantitative measurement of how usable a product is, in the view of the user. It also gives a global measure of usability which includes Efficiency, Affect, Helpfulness, Control and Learn ability, which have been empirically identified as dimensions of perceived usability. SUMI requires at least ten users who have experience of the software to evaluate effectively. A working version of the software must be existing. This is according to Kirakowski J and Corbett M (1993).

The evaluate usability questions were based on five categories which included:

Efficiency: These questions include:

- The speed of this software is fast enough.
- Software at sometime stopped unexpectedly.
- Instruction and prompts are helpful.

Affect: Questions includes:

- Working with this software is satisfying.
- It is obvious that user needs have been fully taken into consideration.
- The software hasn’t always done what I was expecting.
- The software has a very attractive presentation.
This software occasionally behaves in a way which can’t be understood.

**Helpfulness** Questions includes:

- There is never enough information on the screen when it’s needed.
- Either the amount or quality of the help information varies across the system.

**Control:** Questions includes:

- It is easy to see at a glance what the options are at each stage.
- If this software stops, it is not easy to restart it.
- It takes too long to learn the software commands.
- Learning to operate is initially full of problems.
- Sometimes I don't know what to do next with the software.

**Learn ability** Questions includes:

- The way that system information is presented is clear and understandable;
- I can understand and act on the information provided by this software;
- The organization of the menus or information lists seems quite logical;
- I will never learn to use all that is offered in this software.
- I have to look for assistance most times when I use this software.

### 3.8 Limitations of methodology

This study was only limited to the use of Prometheus methodology and JADE framework in demonstrating how an ontology based multiagent system can be used in tender evolution in public institution specifically Government Ministries/departments.
CHAPTER FOR: ANALYSIS AND DESIGN

4.0 Overview
This chapter discusses the analysis and design that will be used in developing the public procurement ontology based multiagent system (PPOBMAS). The chapter is organised as follow: Overview of the existing system, Proposed system, Use case diagram, Agent Diagrams, Class diagram, Entity Relation Diagram (ERD), Database Design and interface design.

4.1 Existing system
The tender evaluation and awarding process in the public service departments entail requirement definition, tender/Quotation advertisement, application for tender/quotation, evaluation criteria, evaluating the bids by considering 30% allocation to special groups and notifying the bidders result. The processes are doing manual which is time consuming, not efficient, costly and not transparent.

Figure 6: Existing system Dataflow diagram
4.2 Proposed system
The proposed system will entail automating the existing system by using multi agents. Agents will be performing various task assigned Agent management system(AMS). The agents will be knowledge based thus reducing human interaction in evaluation of tenders. This section discusses the analysis of Public procurement ontology based multi agent system(PPOBMAS) using use case diagrams and Agent Diagrams which will form the bases of designing the system.

4.2.1 PPOBMAS Use case diagram

![Proposed Use case diagram](image)

**Figure 7: Proposed Use case diagram**
4.2.2 PPOBMAS  Agents System Diagram

Figure 8: Agent system diagram

4.2.3 PPOBMAS  Agents Role Grouping

Figure 9: Agents Role Grouping
<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Responsibilities</th>
</tr>
</thead>
</table>
| Specification Agent    | • Define and store various specification.  
                          • Provide advertisement agent and evaluation criteria agent with specifications.                                                               |
| Advertisement Agent    | • Prepare advert for supplier for online communication.  
                          • Provide response from suppliers.  
                          • Closes the advert.  
                          • Provide evaluation agent with suppliers response.                                                                                           |
| Evaluation criteria Agent | • Get specification from specification agents.  
                               • Determine the criteria to be used in evaluation.  
                               • Provide information to evaluation agent                                                                                                     |
| Evaluation Agent       | • Interact with Evaluation criteria agent.  
                          • Carry technical evaluation.  
                          • Carry financial evaluation.  
                          • Consider 30% allocation to special groups.                                                                                                   |
| Notification Agent     | • Interact with Evaluation agent to get results.  
                          • Notify the bidders result.                                                                                                                    |
| Web Agent              | • Allow staff and consultant/suppliers to interact with the agents i.e. provide an interface with agents.                                                                 |
| Staff Agent            | • Provide the knowledge of the system.                                                                                                             |
| Supplier Agent         | • Respond to the request by the agents.                                                                                                             |
| AMS                    | • Coordinate the agents.                                                                                                                          |
| DF                     | • Agents wishing to advertise their services register with it.                                                                                     |
| Domain Ontology Agent  | • Ensure seamless communication between agents.                                                                                                |

**Table 2: Agents Responsibilities**
4.3 DESIGN

4.3.1 PPOBMAS Hierarchy Class diagram

Figure 10: PPOBMAS Hierarchy Class diagram
4.3.2 Entity Relation Diagram(ERD)

The diagram shows the PPOBMAS entities and their key attributes.

**PPOBMAS**

- Suppliers
  - Names
  - Contacts
  - Category
- Staff
  - Names
  - organisation
- Procurement list
  - Request
  - List Code
  - Listname
- Specification
  - Requirements
  - Evaluation criteria
  - Specification
  - Technical Result
  - Marks
  - List Code
  - Listname
- Evaluation
  - Total Marks
  - Position
- Notification
  - Status
  - Email Address
  - Cellphone

Figure 11: PPOBMAS Entity Relation Diagram(ERD)
4.3.3 Database Design

The following is a generated database relationship of PPOBMAS
CHAPTER FIVE: IMPLEMENTATION

This chapter discusses the implementation process of the prototype and presentation of the results. The chapter is organised as follows: implementation tools used.

5.1 Implementation tools

A number of tools have been used in implementing the system. The tools are as follows:

- **Eclipse** - used to design agents diagrams
- **Java SE** - Java Development Kit (JDK), Server Java Runtime Environment (Server JRE), and Java Runtime Environment (JRE). There library used to run Net beans 8.0.
- **Net beans IDE 8.0** - Which is a powerful integrated development environment for developing applications on the Java platform. Used to write java classes of the system.
- **Jade Framework** - Provide Agent management system (AMS), Directory facilitators (Df) and Remote management Access (RMA). Used as agent development environment
- **Web Service Integrated Gateway (Jetty)** add-on for Jade framework. Used to create web interfaces for users to interact with agents
- **Primefaces 5.0** - development theme libraries, used as template for web portal
- **Databases** – MySQL. Used to design database
- **JPA 2** - Object Relational mapping tool.
- **Client computers** - Installed with Windows 7 professional.

5.2 Agents deployment

5.2.1 Launch of agents
5.2.2 Sniffer Agent

![Figure 12: Launching of agent multiagents system](image1)

Figure 12: Launching of agent multiagents system

5.3 System testing and evaluation

Software testing is an important activity in software development in order to ensure the correctness of software. However, testing is often disregarded in most agent oriented
methodologies because it mainly focuses on analysis and design activities and considered that implementation and testing issues can be performed using traditional techniques.

The testing stage is not at all covered by Prometheus. Because multi-agent systems implementation has some features that make it distinctive from traditional software. Since agent has some peculiar properties like autonomy, creativeness, pro-activeness and social ability, it is not possible to test agent oriented systems using OO testing because objects communicate by method invocation whereas agents communicate by means of message passing.

5.3.1 Testing
The design of how each agent achieves its goals is expressed in terms of plans, events, and data. Thus the final part of detailed design phase develops data, events, and plans. Developing the internal structure of an agent is done using a refinement process that begins with the interface of the agent: the data that it reads and writes, the incoming and outgoing messages, the percepts it receives, and actions it performs. Then, for each part of the agent’s interface, we consider to what capability or plan it connects. In addition to connecting the interface of the agent to internal plans and/or capabilities, additional internal messages and data are introduced as needed. There is a concept of an event which triggers selection of one of some number of identified plans, depending on the situation. If one of these plans is actually never used, then this is likely to indicate an error. The concepts of event and plan, and the relationships between them are part of typical agent designs, and can thus be used for model based testing of agent systems.

Web interface testing
The prototype was first tested within the development environment using virtual webservice where errors and logic flow were corrected and examined before subjecting to end user testing. Testing was also done to ensure that the functionality agents perceive environment according to the specified requirements. The process adhered to principle of Component, integrated, system, performance and evaluation.

5.4 Presentation of results
5.4.1 Agents interactions results
The following agents shows how the achieve a common goal of evaluating a tenders
Figure 14: Web agent interacts with specification agent.

Figure 15: Web agent interacts with evaluation criteria agent.
Figure 16: Web agent interact with advertisement agent

Figure 17: Evaluation agent evaluating and Notifying bidders through Notification Agent
5.4.2 Web agents interactions results

Figure 18: Web Agent evaluation results

5.5 Usability Analysis results
The prototype was deployed and installed on desktop which is connected to GCCN. This ensured the users can access the prototype via the intranet. It was then demonstrated to thirteen (13) officers individually, on the operation of the application. They were then allowed to practically use the application where ICT department provided technical specification and evaluation criteria while supply chain department acted as procuring entity and suppliers/Consultant. This was made to ensure all areas of system were evaluated within the time constraints.

**Efficiency:** This refers to the degree to which the user can achieve the goals of interacting with prototype directly and timely manner.

<table>
<thead>
<tr>
<th>Usability</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Freq</td>
<td>%</td>
<td>Freq</td>
<td>%</td>
</tr>
<tr>
<td>The speed of this software is fast enough.</td>
<td>11</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Software at sometime stopped unexpectedly.</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>18.1</td>
</tr>
<tr>
<td>Instruction and prompts are helpful.</td>
<td>10</td>
<td>90.9</td>
<td>1</td>
<td>9.0</td>
</tr>
</tbody>
</table>

Table 3: Result table for system efficiency
Figure 19: Result shows the efficiency of the system

Affect: Refers to how much the application captures the user’s emotional responses

<table>
<thead>
<tr>
<th>Usability</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working with this software is satisfying.</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>It is obvious that user needs have been fully taken into consideration.</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>The software hasn’t always done what I was expecting.</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>The software has a very attractive presentation.</td>
<td>9</td>
<td>2</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>This software occasionally behaves in a way which can’t be understood.</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4: Result table for Affect

Figure 20: Result shows the affect of the system to the users
**Control:** Is the degree to which the user feels controlling the flow of system

<table>
<thead>
<tr>
<th>Usability</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frequecy</td>
<td>%</td>
<td>Frequecy</td>
<td>%</td>
</tr>
<tr>
<td>It is easy to see at a glance what the options are at each stage.</td>
<td>11</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>If this software stops, it is not easy to restart it.</td>
<td>3</td>
<td>27.27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>It takes too long to learn the software commands</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Learning to operate is initially full of problems.</td>
<td>2</td>
<td>18.18</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sometimes I don't know what to do next with the software</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 5:** Result table for system Control

![Bar chart showing frequency of responses for Control and Helpfulness](chart.png)

**Figure 21:** Result shows the feel of user controlling system

**Helpfulness:** This is the extent to which the application seems to assist the user

<table>
<thead>
<tr>
<th>Usability</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Frequecy</td>
<td>%</td>
<td>Frequecy</td>
<td>%</td>
</tr>
<tr>
<td>There is never enough information on the screen when it’s needed.</td>
<td>2</td>
<td>18.18</td>
<td>1</td>
<td>9.09</td>
</tr>
<tr>
<td>Either the amount or quality of the help information varies across the system.</td>
<td>6</td>
<td>54.55</td>
<td>2</td>
<td>18.1</td>
</tr>
</tbody>
</table>
**Learn ability:** The ease with which a user can get started and learn new features of the product

<table>
<thead>
<tr>
<th>Usability</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The way that system information is presented is clear and understandable</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>I can understand and act on the information provided by this software;</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>The organization of the menus or information lists seems quite logical;</td>
<td>8</td>
<td>72</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>I will never learn to use all that is offered in this software.</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>I have to look for assistance most times when I use this software.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
</tbody>
</table>

*Figure 22: Result shows the helpfulness of system*
Figure 23: Result shows the learn ability of the system
CHAPTER SIX: CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion
The research project undertaken shows that ontology based multiagent system can be used in developing public procurement shared services. This was achieved by use of various agents in achieving tender evaluation goal. The software agents performed their role independently by perceiving its environment and interacting using agent communication language (ACL). Multiagents system have unique features which make it good for distributed system compared to object oriented i.e. autonomy, social ability, reactivity, temporal continuity features.

The study demonstrated how agents can be used in tender evaluation. To achieve the goal, a number of agents were identified to work together by sharing information seamlessly which was enabled by use of ontology's embedded in jade framework. The agents were determined based on the processes undertaken before evaluation can be achieved. The processes included: identification of specification, generation of evaluation criteria, advertisement and response of tenders, technical and financial evaluation, and notification of bidders. Every stage was handled by agent which worked based on the knowledge it has while perceiving the environment. The prototype was evaluated using the business owner and response was positive. The observation was that, transparency, low cost, reduced tender evaluation time and allocation of 30% to youth, women and disabled in tender evaluation is achievable using multiagent system.

6.2 Key Achievements
The study managed to achieve the set objectives. The overall objective was to provide a software agents which can achieve public procurement tender evaluation goals. This was achieved through developing an ontology based multiagents system composed of the following agents: Web agent, Sniffer agent, specification agent, evaluation criteria agent, advertisement agent, evaluation agent and notification agent.

The specific objectives were also achieved: The first was to identify the required factors for the development of the public procurement tender evaluation multiagents system. This was attained in the early stages of the study after reviewing the existing systems and identifying gaps in them and also in collecting information on requirement of tender evaluation. The second objective was to develop a multiagent system framework for e-procurement tender evaluation process. This was developed using Prometheus methodology showing how agents will be working together to achieve their goals. The
third objectives was to develop an ontology-based multiagent system prototype. This was developed and evaluated by users who found it useful in enhancing efficiency of tender evaluation in terms of time period, allocation of resources, cost, transparency and accountability.

6.3 Limitations of the study

During the study there were some challenges that were encountered and some were solved. They included: Acquiring correct first hand information for the application was a challenge. This was solved by reviewing statutory documents, online materials and getting more information from business process owner. The other challenge was getting extra hours to develop the prototype due office workload. This was mitigated by working late in the day and night in order to recover project man hours. Another challenge was getting users to evaluate the application. This was mostly because the system was installed in a desktop machine where some machine of targeted officers are in a different subnets. The study overcame this challenge by requesting the Head of ICT to allow some machine to join the subnet where the prototype is running.

6.4 Recommendations

The study found out that multiagent system can be used in public procurement tender evaluation process this is because of their ability of autonomy, reactive, social ability and temporal continuity. This was evident on how business owner appreciated the efficiency of software agents carrying out evaluation in terms of time taken, volume of documents to evaluate, objectivity of exercise and allocation of resources to special group. They opinion was that the concept has come at the right time when there is transformation agenda on how to procure good and services online. Therefore, the researcher recommends piloting of the application by the Government technical departments to appreciate the use of software agents in tender evaluation and incorporate it in other modules of procurement processes.

More work need to be done to make the system have high intelligent of evaluating complex consultancy and also be able to verify information from other government systems.
REFERENCES


APPENDICES

APPENDIX A: Interview Checklist

1. Job Title?
2. What is your professional?
3. What are the procurement processes are you aware of?
4. In reference to the above, which one are they implemented in the IFMIS?
5. How does your organization ensure 30% allocation of tender/quotation to special groups?
6. When floating a tender/quotation, who prepares Term of Reference/Specification?
7. In terms of working days, how long does it take for suppliers to know the results of tender evaluation?
8. If the above process is computerized, can the period for suppliers to get results of the tender evaluation will be shorter?
9. How is the technical/financial evaluation done in your Ministry/Department?
10. What is the minimum number of officers required when constituting evaluation committee
11. When technical/financial evaluation is being done, are they guided by evaluation criteria?
12. In terms of working days, how long does it take to carry out evaluation of about 10 tenders/quotation?
13. Has there been attempt to automate evaluation process?
14. If no what might be the reason of not automating?
APPENDIX B: Application Usability Evaluation questionnaire.
These questions were adapted from Software Usability Measurement Inventory (SUMI) questionnaire.
The following questions ask about your experience when using the PPOBMAS. Please take few minutes to respond to them. Use options of whether you Agree (1), Undecided (2) or Disagree (3).

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Agree</th>
<th>Undecided</th>
<th>Disagree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The software has at some time stopped unexpectedly.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Learning to operate this software initially is full of problems.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>I enjoy my sessions with this software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>I find that the help information given by this software is not very useful.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Working with this software is satisfying.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>The way that system information is presented is clear and understandable.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>There is never enough information on the screen when it’s needed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>I feel in command of this software when I am using it.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>I would not like to use this software every day.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>I can understand and act on the information provided by this software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>There is too much to read before you can use the software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Tasks can be performed in a straightforward manner using this software.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>The software has helped me overcome any problems I have had in using it.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>The speed of this software is fast enough.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>The organisation of the menus or information lists seems quite logical.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Learning how to use new functions is difficult.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>There are too many steps required to get something to work.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Error prevention messages are not adequate.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>The software hasn’t always done what I was expecting.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>The software has a very attractive presentation.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>It is relatively easy to move from one part of a task to another.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>It have to look for assistance most times when I use this software.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX C: PPOBMAS Sample Code

Project launch
package org.agent.based.tendering;
import jade.core.Profile;
import jade.core.ProfileImpl;
import jade.wrapper.AgentContainer;
import javax.swing.JOptionPane;
import org.agent.based.tendering.agents.PlatformAgent;
import org.agent.based.tendering.jpa.models.ModelManager;
import org.agent.based.tendering.jpa.models.UserRole;
import org.agent.based.tendering.jpa.models.SystemUser;

public static void main(String[] args) {
    try {
        new Project().prepare();
        Profile profile = new ProfileImpl("localhost", 1099,
            Profile.PLATFORM_ID);
        profile.setParameter(Profile.PLATFORM_ID, "Agent Based Tendering Platform");
        profile.setParameter("gui", "true");
        AgentContainer container =
            jade.core.Runtime.instance().createMainContainer(profile);
        container.start();
        container.createNewAgent("Platform Agent", PlatformAgent.class.getName(),
            null).start();
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(null, "Could not start JADE platform", "Start-up
            failed", JOptionPane.ERROR_MESSAGE);
        System.exit(0);
    }
}

private void prepare() {
    SystemUser t = new SystemUser();
    t.setUsername("administrator");
    t.setUserRole(UserRole.ADMIN);
    t.setPassword("password");
    try {
        ModelManager.getInstance().save(t);
    } catch (Exception e) {
    }
}

Agent Abstract
package org.agent.based.tendering.agents;

import jade.core.AID;
import jade.core.Agent;
import jade.domain.DFService;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.SearchConstraints;
import jade.domain.FIPAAgentManagement.ServiceDescription;
public abstract class AbstractAgent extends Agent {

    public final String SNIFFER = "Platform Sniffer";
    public final String SNIFFER_STARTER = "Sniffer Starter";
    public final String WEB_AGENT = "Web Agent";
    public final String SPECIFICATION_AGENT = "Specification Agent";
    public final String ADVERTISEMENT_AGENT = "Advertisement Agent";
    public final String EVALUATION_CRITERIA_AGENT = "Evaluation Criteria Agent";
    public final String EVALUATION_AGENT = "Evaluation Agent";
    public final String NOTIFICATION_AGENT = "Notification Agent";

    protected final org.slf4j.Logger log = LoggerFactory.getLogger(getClass());

    protected void takeDown() {
        log.debug(" takeDown -- closing agent");
        try {
            DFService.deregister(this);
        } catch (FIPAException ex) {
            log.error(" takeDown -- error ", ex);
        }
        try {
            super.takeDown(); //To change body of generated methods, choose Tools | Templates.
        } catch (Exception e) {
            log.error(" takeDown -- error ", e);
        }
    }

    protected void setup() {
        super.setup(); //To change body of generated methods, choose Tools | Templates.
        log.debug(" setup -- initializing agent {} ", getLocalName());
        register();
    }

    protected final void register() {
        log.debug(" register -- registering agent {} with DF ", getLocalName());
        DFAgentDescription dfd = new DFAgentDescription();
        dfd.setName(getAID());
        ServiceDescription sd = new ServiceDescription();
        sd.setType(getClass().getSimpleName());
        sd.setName(getLocalName());
        dfd.addServices(sd);
        try {
            DFService.register(this, dfd);
        } catch (FIPAException fe) {
            log.error(" register -- error ", fe);
        }
    }
}

import jade.domain.FIPAException;
import jade.lang.acl.ACLMessage;
import java.io.IOException;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.agent.based.tendering.jpa.models.Notification;
import org.slf4j.LoggerFactory;
protected final AID getService(String service) {
    log.debug(" getServices -- searching DF for one {} ", service);
    DFAgentDescription dfd = new DFAgentDescription();
    ServiceDescription sd = new ServiceDescription();
    sd.setType(service);
    dfd.addServices(sd);
    try {
        DFAgentDescription[] result = DFService.search(this, dfd);
        if (result.length > 0) {
            return result[0].getName();
        }
    } catch (FIPAException fe) {
        log.error(" get Service -- error ", fe);
    }
    return null;
}

protected final AID[] getServices(String service) {
    log.debug(" get Services -- searching DF for {} ", service);
    DFAgentDescription dfd = new DFAgentDescription();
    ServiceDescription sd = new ServiceDescription();
    sd.setType(service);
    dfd.addServices(sd);
    SearchConstraints ALL = new SearchConstraints();
    ALL.setMaxResults(new Long(-1));
    try {
        DFAgentDescription[] result = DFService.search(this, dfd, ALL);
        AID[] agents = new AID[result.length];
        for (int i = 0; i < result.length; i++) {
            agents[i] = result[i].getName();
        }
        return agents;
    } catch (FIPAException fe) {
        log.error(" search DF -- error ", fe);
    }
    return null;
}

public void handleDatabaseChanges(Serializable contentObject) {
}

public void sendNotification(Notification n) {
    try {
        ACLMessage am = new ACLMessage(ACLMessage.REQUEST);
        am.addReceiver(new AID(NOTIFICATION_AGENT, AID.ISLOCALNAME));
        am.setContentObject(n);
    } catch (FIPAException fe) {
        log.error(" send Notification -- error ", fe);
    }
}
import jade.core.AID;
import jade.core.behaviours.CyclicBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import java.io.Serializable;
import org.agent.based.tendering.agents.extras.HandleDBRequests;
import org.agent.based.tendering.jpa.models.Advertisement;
import org.agent.based.tendering.jpa.models.ModelManager;
import org.agent.based.tendering.jpa.models.Tender;
public class AdvertisementAgent extends AbstractAgent {
    @Override
    public void handleDatabaseChanges(Serializable contentObject) {
        if (contentObject == null) {
            return;
        }
        WebAgent.EW ew = (WebAgent.EW) contentObject;
        contentObject = ew.getEntity();
        if (contentObject instanceof Tender) {
            Advertisement ad = new Advertisement();
            Tender t = (Tender) contentObject;
            ad.setTender(t);
            ad.setTenderNo(t.getTenderNo());
            ad.setClosingDate(t.getEndDate());
            ad.setOpeningDate(t.getStartDate());
            ad.setItems(t.getItems());
            ModelManager.getInstance().save(ad, true);
            t.setStage(org.agent.based.tendering.jpa.models.ProcessingStage.PICKED);
            ModelManager.getInstance().save(t, true);
        }
    }
    protected void setup() {
        super.setup(); //To change body of generated methods, choose Tools | Templates.
        addBehaviour(new HandleDBRequests(this));
        addBehaviour(new Advertise());
    }
}

public void action() {
    ACLMessage receive = receive(m);
    if (receive != null) {

    }
    block();
}

**Evaluation agent**

package org.agent.based.tendering.agents;

import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.TickerBehaviour;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import org.agent.based.tendering.agents.extras.HandleDBRequests;
import org.agent.based.tendering.jpa.models.Advertisement;
import org.agent.based.tendering.jpa.models.EvaluationCriteria;
import org.agent.based.tendering.jpa.models.EvaluationLog;
import org.agent.based.tendering.jpa.models.ModelManager;
import org.agent.based.tendering.jpa.models.Notification;
import org.agent.based.tendering.jpa.models.ProcessingStage;
import org.agent.based.tendering.jpa.models.ProductionItem;
import org.agent.based.tendering.jpa.models.Specification;
import org.agent.based.tendering.jpa.models.Supplier;
import org.agent.based.tendering.jpa.models.SupplierAdFlag;
import org.agent.based.tendering.jpa.models.SupplierEvaluation;
import org.agent.based.tendering.jpa.models.SupplierSpecification;
import org.agent.based.tendering.jpa.models.SupplierSpecificationScore;
import org.agent.based.tendering.jpa.models.Tender;
import org.agent.based.tendering.jpa.models.TenderCost;
import org.agent.based.tendering.utils.ResponseManager;
import org.joda.time.DateTime;

public class EvaluationAgent extends AbstractAgent {
    private ResponseManager rm;
    public EvaluationAgent() {
        rm = new ResponseManager();
    }
    @Override
}
protected void setup() {
    super.setup(); //To change body of generated methods, choose Tools | Templates.
    addBehaviour(new HandleDBRequests(this));
    addBehaviour(new EvaluationChecker(15l));
}

private void notifyAwarded(SupplierEvaluation supplierEvaluation, ProcurementItem item) {
    EvaluationLog el = new EvaluationLog();
    el.setAdvertisement(supplierEvaluation.getTender().getAdvertisement());
    el.setComments("Awarded");
    el.setAwarded(Boolean.TRUE);
    el.setFinancial(supplierEvaluation.getFinancialResult());
    el.setTechnical(supplierEvaluation.getTechnicalResult());
    el.setPosition(supplierEvaluation.getPosition());
    el.setItem(item);
    el.setSupplier(supplierEvaluation.getSupplier());
    ModelManager.getInstance().save(el);
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    Map<String, String> map = new HashMap<>();
    map.put("company", supplierEvaluation.getSupplier().getName());
    map.put("tender", supplierEvaluation.getTender().getTenderNo());
    map.put("item", item.getListCode() + " " + item.getListName());
    String htmlMessage = rm.getHTMLMessage(ResponseManager.AWARDED, map);
    Notification n = new Notification();
    n.setEmail(supplierEvaluation.getSupplier().getEmail());
    n.setTender(supplierEvaluation.getTender());
    n.setSupplier(supplierEvaluation.getSupplier());
    n.setMessage(htmlMessage);
    n.setStatus("OK");
    ModelManager.getInstance().save(n, true);
    sendNotification(n);
}

private void notifyNotAwarded(List<SupplierEvaluation> selected, ProcurementItem item, SupplierEvaluation awardedTo) {
    for (SupplierEvaluation supplierEvaluation : selected) {
        EvaluationLog el = new EvaluationLog();
        el.setAdvertisement(awardedTo.getTender().getAdvertisement());
        el.setComments("A different supplier offered the more acceptable terms");
        el.setAwarded(Boolean.FALSE);
        el.setFinancial(supplierEvaluation.getFinancialResult());
        el.setTechnical(supplierEvaluation.getTechnicalResult());
        el.setItem(item);
        el.setSupplier(supplierEvaluation.getSupplier());
        ModelManager.getInstance().save(el, true);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Map<String, String> map = new HashMap<>();
        map.put("company", supplierEvaluation.getSupplier().getName());
        map.put("awarded", awardedTo.getSupplier().getName());
        map.put("tender", supplierEvaluation.getTender().getTenderNo());
    }
}
map.put("item", item.getListCode() + " " + item.getListName());
String htmlMessage =
rm.getHTMLMessage(ResponseManager.NOT_AWARDED, map);
Notification n = new Notification();
n.setEmail(supplierEvaluation.getSupplier().getEmail());
n.setTender(supplierEvaluation.getTender());
n.setSupplier(supplierEvaluation.getSupplier());
n.setMessage(htmlMessage);
n.setStatus("OK");
ModelManager.getInstance().save(n, true);
sendNotification(n);
}
private void notifyNoneAwarded(List<SupplierEvaluation> selected, ProcurementItem item) {
    for (SupplierEvaluation supplierEvaluation : selected) {
        EvaluationLog el = new EvaluationLog();
el.setAdvertisement(item.getTender().getAdvertisement());
el.setComments("No supplier offered acceptable terms");
el.setAwarded(Boolean.FALSE);
el.setFinancial(supplierEvaluation.getFinancialResult());
el.setTechnical(supplierEvaluation.getTechnicalResult());
el.setItem(item);
el.setPosition(supplierEvaluation.getPosition());
el.setSupplier(supplierEvaluation.getSupplier());
ModelManager.getInstance().save(el);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Map<String, String> map = new HashMap<>();
map.put("company", supplierEvaluation.getSupplier().getName());
map.put("tender", supplierEvaluation.getTender().getTenderNo());
map.put("item", supplierEvaluation.getItem().getListCode() + " " +
supplierEvaluation.getItem().getListName());
String htmlMessage =
rm.getHTMLMessage(ResponseManager.NONE_AWARDED, map);
Notification n = new Notification();
n.setEmail(supplierEvaluation.getSupplier().getEmail());
n.setTender(supplierEvaluation.getTender());
n.setSupplier(supplierEvaluation.getSupplier());
n.setMessage(htmlMessage);
n.setStatus("OK");
ModelManager.getInstance().save(n, true);
sendNotification(n);
}
private void notifyDisqualifiedBySpecialGroup(List<SupplierEvaluation> selected, ProcurementItem item) {
    for (SupplierEvaluation supplierEvaluation : selected) {
        EvaluationLog el = new EvaluationLog();
el.setAdvertisement(supplierEvaluation.getTender().getAdvertisement());

el.setComments("Disqualified by special group consideration.");
el.setAwarded(Boolean.FALSE);
el.setFinancial(supplierEvaluation.getFinancialResult());
el.setTechnical(supplierEvaluation.getTechnicalResult());
el.setPosition(supplierEvaluation.getPosition());
el.setItem(item);
el.setSupplier(supplierEvaluation.getSupplier());
ModelManager.getInstance().save(el);
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Map<String, String> map = new HashMap<>();
map.put("company", supplierEvaluation.getSupplier().getName());
map.put("tender", supplierEvaluation.getTender().getTenderNo());
map.put("item", supplierEvaluation.getItem().getListCode() + " " +
supplierEvaluation.getItem().getListName());
String htmlMessage =
rm.getHTMLMessage(ResponseManager.NOT_AWARDED_SPECIAL_GROUP, map);
Notification n = new Notification();
n.setEmail(supplierEvaluation.getSupplier().getEmail());
n.setTender(supplierEvaluation.getTender());
n.setSupplier(supplierEvaluation.getSupplier());
n.setMessage(htmlMessage);
n.setStatus("OK");
ModelManager.getInstance().save(n, true);
sendNotification(n);
}
private void notifyNotSubmittedBeforedDeadline(SupplierAdFlag supplierAdFlag) {
try {
    for (ProcurementItem procurementItem :
supplierAdFlag.getAdvertisement().getItems()) {
        EvaluationLog el = new EvaluationLog();
el.setAdvertisement(supplierAdFlag.getAdvertisement());
el.setComments("Application not completed in time.");
el.setAwarded(Boolean.FALSE);
el.setFinancial(null);
el.setItem(procurementItem);
el.setTechnical(0d);
el.setSupplier(supplierAdFlag.getSupplier());
ModelManager.getInstance().save(el);
    }
} catch (Exception e) {
}
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Map<String, String> map = new HashMap<>();
map.put("company", supplierAdFlag.getSupplier().getName());
map.put("tender", supplierAdFlag.getAdvertisement().getTenderNo());
map.put("opening",
sdf.format(supplierAdFlag.getAdvertisement().getOpeningDate()));
map.put("closing",
sdf.format(supplierAdFlag.getAdvertisement().getClosingDate()));
String htmlMessage =
rm.getHTMLMessage(ResponseManager.INCOMPLETE_APPLICATION, map);
Notification n = new Notification();
n.setEmail(supplierAdFlag.getSupplier().getEmail());
n.setTender(supplierAdFlag.getAdvertisement().getTender());
n.setSupplier(supplierAdFlag.getSupplier());
n.setMessage(htmlMessage);
n.setStatus("OK");
ModelManager.getInstance().save(n, true);
sendNotification(n);
}

private void notifyFailedTechnicalEvaluation(SupplierAdFlag supplierAdFlag, double score, ProcurementItem item) {
EvaluationLog el = new EvaluationLog();
el.setAdvertisement(supplierAdFlag.getAdvertisement());
el.setComments("Failed technical evaluation");
el.setAwarded(Boolean.FALSE);
el.setTechnical(score);
el.setItem(item);
el.setSupplier(supplierAdFlag.getSupplier());
ModelManager.getInstance().save(el);
Map<String, String> map = new HashMap<>();
map.put("company", supplierAdFlag.getSupplier().getName());
map.put("tender", supplierAdFlag.getAdvertisement().getTenderNo());
map.put("score", "+ displayNumber(score * 100d / 80d) + "/\%");
map.put("item", "+ item.getListCode() + "/\"");
map.put("passmark", "+ displayNumber(70d) + "/\%");
String htmlMessage =
rm.getHTMLMessage(ResponseManager.FAILED_TECH_ANALYSIS, map);
Notification n = new Notification();
n.setEmail(supplierAdFlag.getSupplier().getEmail());
n.setTender(supplierAdFlag.getAdvertisement().getTender());
n.setSupplier(supplierAdFlag.getSupplier());
n.setMessage(htmlMessage);
n.setStatus("OK");
ModelManager.getInstance().save(n, true);
sendNotification(n);
}

public String displayNumber(Double d) {
if (d == null) {
    return "N/A";
}
if (d == 0) {
    return "0.00";
}
if (d * 100d % 100 == 0) {
    return d.intValue() + "\"";
}
DecimalFormat df = new DecimalFormat("#0.00");
private class Evaluate extends OneShotBehaviour {

    private final List<SupplierAdFlag> flags;

    public Evaluate(List<SupplierAdFlag> flags) {
        super(EvaluationAgent.this);
        this.flags = flags;
    }

    @Override
    public void action() {
        if (flags.isEmpty()) {
            log.warn("Evaluate.action -- no applications to process");
            return;
        }
        Advertisement ad = flags.get(0).getAdvertisement();
        Map<ProcurementItem, List<SupplierEvaluation>> scores = new HashMap<>();
        for (ProcurementItem procurementItem : ad.getItems()) {
            List<SupplierEvaluation> evaluateItem = evaluateItem(procurementItem);
            scores.put(procurementItem, evaluateItem);
        }
        // we got technical and financial scores for all
        for (ProcurementItem procurementItem : scores.keySet()) {
            List<SupplierEvaluation> scoreList = scores.get(procurementItem);
            class MSP {
                SupplierEvaluation s;
                double score;
                int position;
            }
            List<MSP> order = new ArrayList<>();
            List<MSP> ordered = new ArrayList<>();
            for (SupplierEvaluation supplierEvaluation : scoreList) {
                Double tech = supplierEvaluation.getTechnicalResult();
                Double fin = supplierEvaluation.getFinancialResult();
                MSP m = new MSP();
                m.s = supplierEvaluation;
                m.score = tech + fin;
                order.add(m);
            }
            int pos = 1;
            while (!order.isEmpty()) {
                double max = 0;
                // get maximum, to assign highest position
                for (MSP msp : order) {
                    if (msp.score > max) {
                        max = msp.score;
                    }
                }
                ordered.add(pos, order.get(order.size() - 1 - pos++));
            }
        }
    }
}
int ties = 0;
for (MSP msp : order) {
    if (msp.score == max) {
        msp.position = pos;
        msp.s.setPosition(pos);
        ordered.add(msp);
        ties++;
    }
}
for (MSP msp : ordered) {
    try {
        order.remove(msp);
    } catch (Exception e) {
        pos += ties;
    }
}
List<SupplierEvaluation> evalutaions = new ArrayList<>();
for (MSP msp : ordered) {
    ModelManager.getInstance().save(msp.s, true);
    evalutaions.add(msp.s);
}
chooseSupplier(evalutaions, procurementItem);
}
ad.setStage(ProcessingStage.FINISHED);
ModelManager.getInstance().save(ad, true);
}
private List<SupplierEvaluation> evaluateItem(ProcurementItem procurementItem) {
    String s = "SELECT a FROM EvaluationCriteria a WHERE a.item = :item";
    Map<String, Object> map = new HashMap<>();
    map.put("item", procurementItem);
    List<EvaluationCriteria> criteria = ModelManager.getReadingInstance().findEntities(s, map);
    double total = 0d;
    for (EvaluationCriteria evaluationCriteria : criteria) {
        total += evaluationCriteria.getMarks();
    }
    Pattern numbers = Pattern.compile("[0-9]+(\.[0-9]+)?");
    Set<SupplierSpecificationScore> scores = new HashSet<>();
    for (EvaluationCriteria evaluationCriteria : criteria) {
        Matcher m1 = numbers.matcher(evaluationCriteria.getRequired());
        if (m1.find()) {
            //check numbers
            String group = m1.group();
            double evaluator = Double.parseDouble(group);
            for (SupplierAdFlag supplierAdFlag : this.flags) {
                SupplierSpecification sp = getSupplierSpec(supplierAdFlag.getSupplier(),
                                                        evaluationCriteria.getSpecification());
                SupplierSpecificationScore si = new SupplierSpecificationScore();
                si.setAdFlag(supplierAdFlag);
                si.setSpecification(evaluationCriteria.getSpecification());
            }
        }
    }
}
Matcher matcher = numbers.matcher(sp.getProviding());
if (matcher.find()) {
    //number specification expected
    String group1 = matcher.group();
    double providing = Double.parseDouble(group1);
    Double get = 0d;

    String required = evaluationCriteria.getRequired().replace(group, "").toLowerCase().replace( "\", "");
    String replace = sp.getProviding().replace(group1, "").toLowerCase().replace( "\", "");
    if (providing >= evaluator) {
        //check units
        if (replace.equals(required)) {
            si.setComments("OK");
            sp.setComment(si.getComments());
            get += 80d * evaluationCriteria.getMarks() / total;
        } else {
            sp.setComment("Unacceptable units");
            si.setComments(sp.getComment());
        }
    } else {
        if (replace.equals(required)) {
            si.setComments("Value below required");
        } else {
            sp.setComment("Unacceptable units");
            si.setComments(sp.getComment());
        }
    }
    si.setScore(get);
} else {
    //number specification expected, not found
    si.setScore(0d);
    sp.setComment("Unacceptable");
    si.setComments(sp.getComment());
}
si.setMax(80d * evaluationCriteria.getMarks() / total);
ModelManager.getInstance().save(si, true);
scores.add(si);
} else {
    //compare strings
    for (SupplierAdFlag supplierAdFlag : flags) {
        SupplierSpecificationScore si = new SupplierSpecificationScore();
        si.setAdFlag(supplierAdFlag);
        si.setSpecification(evaluationCriteria.getSpecification());
        si.setMax(80d * evaluationCriteria.getMarks() / total);
        Double get = 0d;
        SupplierSpecification sp = getSupplierSpec(supplierAdFlag.getSupplier(), evaluationCriteria.getSpecification());

if (evaluationCriteria.getRequired().toLowerCase().replace(" ", ",").contains(sp.getProviding().toLowerCase().trim().replace(" ", ","))) {
    get += 80d * evaluationCriteria.getMarks() / total;
    si.setComments("OK");
    sp.setComment(si.getComments());
} else {
    si.setComments("Value not as expected");
}

si.setScore(get);
ModelManager.getInstance().save(si, true);
scores.add(si);

Map<SupplierAdFlag, Double> technicalscores = new HashMap<>();
for (SupplierSpecificationScore supplierSpecificationScore : scores) {
    Double get = technicalscores.get(supplierSpecificationScore.getAdFlag());
    if (get == null) {
        get = 0d;
    }
    get += supplierSpecificationScore.getScore();
    technicalscores.put(supplierSpecificationScore.getAdFlag(), get);
}

//financial analysis
Map<SupplierAdFlag, TenderCost> adTenderCosts =
getAdTenderCosts(procurementItem);
double lowest = Double.MAX_VALUE;
int passedFinancialEvaluation = 0;
for (TenderCost tenderCost : adTenderCosts.values()) {
    Double get = technicalscores.get(tenderCost.getFlag());
    if (get < 70d * 80d / 100d) {
        //failed to reach 70% passmark
        //does not get financial evaluation
        //give financial score of -1
        continue;
    }
    passedFinancialEvaluation++;
    if (tenderCost.getUnitPrice() < lowest) {
        lowest = tenderCost.getUnitPrice();
    }
}

List<SupplierEvaluation> evaluations = new ArrayList<>();
for (SupplierAdFlag supplierAdFlag : adTenderCosts.keySet()) {
    Double get = technicalscores.get(supplierAdFlag);
    double finScore = -1;
    if (passedFinancialEvaluation == 0) {
        //failed to reach 70% passmark
        //does not get financial evaluation
        //give financial score of -1
        notifyFailedTechnicalEvaluation(supplierAdFlag, get, procurementItem);
    } else if (get < 70d * 80d / 100d) {
        //failed to reach 70% passmark
    } else {
        evaluations.add(....); //add supplier evaluation
    }
}

//additional code...
// does not get financial evaluation
// give financial score of -1
notifyFailedTechnicalEvaluation(supplierAdFlag, get, procurementItem);
}
else {
    TenderCost tenderCost = adTenderCosts.get(supplierAdFlag);
    finScore = 20d * lowest / tenderCost.getUnitPrice();
    SupplierEvaluation sp = new SupplierEvaluation();
    sp.setFinancialResult(finScore);
    sp.setTechnicalResult(get);
    sp.setItem(procurementItem);
    sp.setPosition(null);
    sp.setSupplier(supplierAdFlag.getSupplier());
    sp.setTender(supplierAdFlag.getAdvertisement().getTender());
    sp.setTotal(sp.getTechnicalResult() + sp.getFinancialResult());
    sp.setStatus("Completed");
    ModelManager.getInstance().save(sp);
    evaluations.add(sp);
}
return evaluations;

private SupplierSpecification getSupplierSpec(Supplier supplier, Specification specification) {
    String s = "SELECT s FROM SupplierSpecification s WHERE s.supplier = :supplier AND s.specification = :spec ";
    Map<String, Object> map = new HashMap<>();
    map.put("supplier", supplier);
    map.put("spec", specification);
    List<SupplierSpecification> retu = ModelManager.getReadingInstance().findEntities(s, map);
    return retu.get(0);
}

private Map<SupplierAdFlag, TenderCost> getAdTenderCosts(ProcurementItem item) {
    String s = "SELECT s FROM TenderCost s WHERE s.item = :item AND s.flag.flag = :submitted ";
    Map<String, Object> map = new HashMap<>();
    map.put("item", item);
    map.put("submitted", SupplierAdFlag.FLAG.SUBMITTED);
    List<TenderCost> retu = ModelManager.getReadingInstance().findEntities(s, map);
    Map<SupplierAdFlag, TenderCost> map1 = new HashMap<>();
    for (TenderCost tenderCost : retu) {
        map1.put(tenderCost.getFlag(), tenderCost);
    }
    return map1;
}

private class EvaluationChecker extends TickerBehaviour {

    public EvaluationChecker(long s) {

super(EvaluationAgent.this, s * 1000l);
}

protected void onTick() {
    String s = "SELECT a FROM Advertisement a WHERE a.closingDate < :now AND a.stage = :stage";
    Map<String, Object> map = new HashMap<>();
    // expired yesterday
    map.put("now", DateTime.now().toDateMidnight().toDate());
    //            map.put("now", DateTime.now().plusDays(1).toDateMidnight().toDate());
    map.put("stage", ProcessingStage.NEW);
    List<Advertisement> ads = ModelManager.getReadingInstance().findEntities(s, map);
    for (Advertisement advertisement : ads) {
        s = "SELECT a FROM SupplierAdFlag a WHERE a.advertisement = :ad AND a.flag != :submitted";
        map.clear();
        map.put("ad", advertisement);
        map.put("submitted", SupplierAdFlag.FLAG.SUBMITTED);
        List<SupplierAdFlag> flags = ModelManager.getReadingInstance().findEntities(s, map);
        s = "SELECT a FROM SupplierAdFlag a WHERE a.advertisement = :ad AND a.flag = :submitted";
        List<SupplierAdFlag> toProcess = ModelManager.getReadingInstance().findEntities(s, map);
        for (SupplierAdFlag supplierAdFlag : flags) {
            notifyNotSubmittedBeforedDeadline(supplierAdFlag);
        }
        new Evaluate(toProcess).action();
        advertisement.setStage(ProcessingStage.PICKED);
        ModelManager.getInstance().save(advertisement);
        addBehaviour(new Evaluate(toProcess));
    }
}

private boolean allSpecificationsGiven(SupplierAdFlag af) {
    Advertisement ad = af.getAdvertisement();
    Supplier supplier = af.getSupplier();
    List<SupplierSpecification> supplierSpecifications = getSupplierSpecification(ad.getTender(), supplier);
    for (ProcurementItem procurementItem : ad.getItems()) {
        String s = "SELECT a FROM EvaluationCriteria a WHERE a.item = :item";
        Map<String, Object> map = new HashMap<>();
        map.put("item", procurementItem);
        List<EvaluationCriteria> criteria = ModelManager.getReadingInstance().findEntities(s, map);
        for (EvaluationCriteria criterion : criteria) {
            boolean found = false;
            for (SupplierSpecification supplierSpecification : supplierSpecifications) {
                if (criterion.getSpecification().equals(supplierSpecification.getSpecification())) {
                    found = true;
                    break;
                }
            }
            if (!found) {
                return false;
            }
        }
    }
    return true;
}
found = true;
}
}
if (!found) {
    return found;
}
return true;

private List<SupplierSpecification> getSupplierSpecification(Tender tender, Supplier supplier) {
    List<SupplierSpecification> l = new LinkedList<>();
    try {
        String s = "SELECT s FROM SupplierSpecification s WHERE s.supplier = :supplier AND s.tender = :tender";
        Map<String, Object> map = new HashMap<>();
        map.put("supplier", supplier);
        map.put("tender", tender);
        return ModelManager.getReadingInstance().findEntities(s, map);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return l;
}

private void test() {
    EvaluationChecker evaluationChecker = new EvaluationChecker(2);
    evaluationChecker.action();
    // getSpecialGroupAwardings();
}

public static void main(String[] args) {
    new EvaluationAgent().test();
}

private void chooseSupplier(List<SupplierEvaluation> evaluations, ProcurementItem item) {
    List<SupplierEvaluation> disqualifiedBySpecialGroupCriteria = new ArrayList<>();
    if (item.isConsiderSpecialGroup()) {
        Integer[] specialGroupAwardings = getSpecialGroupAwardings();
        Integer special = specialGroupAwardings[0];
        Integer nonSpecial = specialGroupAwardings[1];
        if (special == 0 && nonSpecial == 0) {
            //do nothing
        } else {
            double newPercentageSpecial = (1d + special.doubleValue()) * 100d / (1d + special.doubleValue() + nonSpecial.doubleValue());
            double newPercentageNonSpecial = (1d + nonSpecial.doubleValue()) * 100d / (1d + special.doubleValue() + nonSpecial.doubleValue());
            for (SupplierEvaluation supplierEvaluation : evaluations) {
                disqualifiedBySpecialGroupCriteria.add(supplierEvaluation);
            }
        }
    }
}
evaluations.clear();
if (newPercentageSpecial > 70d) {
    // 30% quota must be respected
    // assign to non-special
    for (SupplierEvaluation supplierEvaluation :
        disqualifiedBySpecialGroupCriteria) {
        if (!supplierEvaluation.getSupplier().belongsToSpecialGroup()) {
            evaluations.add(supplierEvaluation);
        }
    }
}
if (newPercentageNonSpecial > 70d) {
    // 30% quota must be respected
    // assign to special
    for (SupplierEvaluation supplierEvaluation :
        disqualifiedBySpecialGroupCriteria) {
        if (supplierEvaluation.getSupplier().belongsToSpecialGroup()) {
            evaluations.add(supplierEvaluation);
        }
    }
}
for (SupplierEvaluation e : evaluations) {
    // remove those qualified from disqualified
    disqualifiedBySpecialGroupCriteria.remove(e);
}
if (evaluations.isEmpty()) {
    // no one qualified
    notifyDisqualifiedBySpecialGroup(disqualifiedBySpecialGroupCriteria, item);
    notifyNoneAwarded(disqualifiedBySpecialGroupCriteria, item);
    return;
}
notifyDisqualifiedBySpecialGroup(disqualifiedBySpecialGroupCriteria, item);
// check for monopoly
SupplierEvaluation selected = null;
for (SupplierEvaluation supplierEvaluation : evaluations) {
    if (selected == null) {
        selected = supplierEvaluation;
        continue;
    }
    if (supplierEvaluation.getSupplier().getAwarded() <
        selected.getSupplier().getAwarded() ||
        (Objects.equals(supplierEvaluation.getSupplier().getAwarded(),
        selected.getSupplier().getAwarded()) &&
        supplierEvaluation.getTotal() >
        selected.getTotal())) {
        selected = supplierEvaluation;
    } else {
        if (supplierEvaluation.getPosition() < selected.getPosition()) {
            selected = supplierEvaluation;
        }
    }
}
if (selected == null) {
    //no one qualified
    notifyNoneAwarded(evaluations, item);
    return;
} else {
    selected.getSupplier().incrementAwards();
    item.setAwardedTo(selected.getSupplier());
    ModelManager.getInstance().save(item, true);
    ModelManager.getInstance().save(selected.getSupplier(), true);
    evaluations.remove(selected);
    notifyAwarded(selected, item);
    notifyNotAwarded(evaluations, item, selected);
}

private Integer[] getSpecialGroupAwardings() {
    String s = "Select count(p) From ProcurementItem p";
    Map<String, Object> map = new HashMap<>();
    int total = ModelManager.getReadingInstance().count(s, map);
    s = "Select count(p) From ProcurementItem p WHERE p.awardedTo.specialGroup IS NULL";
    int nonSpecial = ModelManager.getReadingInstance().count(s, map);
    s = "Select count(p) From ProcurementItem p WHERE p.awardedTo.specialGroup IS NOT NULL";
    int special = ModelManager.getReadingInstance().count(s, map);
    return new Integer[]{special, nonSpecial, total};
}

Evaluation Criteria Agent
package org.agent.based.tendering.agents;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.agent.based.tendering.agents.extras.HandleDBRequests;
import org.agent.based.tendering.jpa.models.EvaluationCriteria;
import org.agent.based.tendering.jpa.models.ModelManager;
import org.agent.based.tendering.jpa.models.ModelTemplate;
import org.agent.based.tendering.jpa.models.ProcurementItem;

public class EvaluationCriteriaAgent extends AbstractAgent {
@Override
public void handleDatabaseChanges(Serializable contentObject) {
}

protected void setup() {
    super.setup(); //To change body of generated methods, choose Tools | Templates.
    addBehaviour(new HandleDBRequests(this));
    addBehaviour(new StandardizeCriteria());
}

private class StandardizeCriteria extends CyclicBehaviour {

    private MessageTemplate m = MessageTemplate.MatchSender(new AID(SPECIFICATION_AGENT, AID.ISLOCALNAME));

    public void action() {
        ACLMessage receive = receive(m);
        if (receive != null) {
            try {
                Serializable contentObject = receive.getContentObject();
                ProcurementItem p = (ProcurementItem) contentObject;
                Map<String, Object> map = new HashMap<>();
                String q = "SELECT s FROM EvaluationCriteria s WHERE s.specification.item = :item";
                map.put("item", p);
                List<EvaluationCriteria> cs = ModelManager.getReadingInstance().findEntities(q, map);
                double total = 0;
                for (EvaluationCriteria evaluationCriteria : cs) {
                    total += evaluationCriteria.getMarks();
                }

                if (total != 100d) {
                    for (EvaluationCriteria evaluationCriteria : cs) {
                        evaluationCriteria.setMarks(evaluationCriteria.getMarks() * 100d / total);
                        ModelManager.getInstance().save(evaluationCriteria);
                    }
                }
            } catch (UnreadableException ex) {
                Logger.getLogger(EvaluationCriteriaAgent.class.getName()).log(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                Logger.getLogger(EvaluationCriteriaAgent.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    ACLMessage ac = new ACLMessage(ACLMessage.REQUEST_WHEN);
    ac.setContentObject(p);
    ac.addReceiver(new AID(ADVERTISEMENT_AGENT, AID.ISLOCALNAME));
    send(ac);
}
import jade.core.AID;
import jade.core.behaviours.CyclicBehaviour;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import java.io.Serializable;
import org.agent.based.tendering.agents.extras.HandleDBRequests;
import org.agent.based.tendering.jpa.models.ModelManager;
import org.agent.based.tendering.utils.SMTP;

public class NotificationAgent extends AbstractAgent {

    public void handleDatabaseChanges(Serializable contentObject) {
    }

    protected void setup() {
        super.setup(); //To change body of generated methods, choose Tools | Templates.
        addBehaviour(new HandleDBRequests(this));
        addBehaviour(new HandleEmailRequests(this));
    }

    private class HandleEmailRequests extends CyclicBehaviour {

        private final AbstractAgent agent;
        private final MessageTemplate messageTemplate;

        public HandleEmailRequests(AbstractAgent agent) {
            super(agent);
            this.agent = agent;
            messageTemplate = MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
        }

        @Override
        public void action() {
            ACLMessage receive = agent.receive(messageTemplate);
            if (receive != null) {
                try {
                    Notification n = (Notification) receive.getContentObject();
                    SMTP.getInstance().sendEmail(n.getEmail(), "Tender status", n.getMessage());
                    n.setStage(org.agent.based.tendering.jpa.models.ProcessingStage.SUCCESS);
                    ModelManager.getInstance().save(n);
                } catch (Exception e) {
                    log.error(" HandleEmailRequests.action -- ", e);
                }
            }
        }
    }
}
Specification agent
package org.agent.based.tendering.agents;
import jade.core.AID;
import jade.lang.acl.ACLMessage;
import java.io.IOException;
import java.io.Serializable;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.agent.based.tendering.agents.extras.HandleDBRequests;
import org.agent.based.tendering.jpa.models.ProcurementItem;

public class SpecificationAgent extends AbstractAgent {
    public void handleDatabaseChanges(Serializable contentObject) {
        if (contentObject == null) {
            return;
        }
        WebAgent.EW ew = (WebAgent.EW) contentObject;
        contentObject = ew.getEntity();
        if (contentObject instanceof ProcurementItem) {
            ProcurementItem p = (ProcurementItem) contentObject;
            if (p.getStage() ==
                org.agent.based.tendering.jpa.models.ProcessingStage.PICKED) {
                try {
                    //forward to evaluation criteria agent to standardize evaluation scores
                    ACLMessage a = new ACLMessage(ACLMessage.REQUEST);
                    a.addReceiver(new AID(EVALUATION_CRITERIA_AGENT, AID.ISLOCALNAME));
                    a.setContentObject(contentObject);
                    send(a);
                } catch (IOException ex) {
                    Logger.getLogger(SpecificationAgent.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        }
    }

    protected void setup() {
        super.setup(); //To change body of generated methods, choose Tools | Templates.
        addBehaviour(new HandleDBRequests(this));
    }
}

package org.agent.based.tendering.agents;
import jade.core.AID;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.TickerBehaviour;
import jade.lang.acl.ACLMessage;
import java.io.IOException;
import java.io.Serializable;
import java.lang.management.ManagementFactory;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.agent.based.tendering.jpa.models.EvaluationCriteria;
import org.agent.based.tendering.jpa.models.ModelTemplate;
import org.agent.based.tendering.jpa.models.Notification;
import org.agent.based.tendering.jpa.models.ProcurementItem;
import org.agent.based.tendering.jpa.models.Specification;
import org.agent.based.tendering.jpa.models.SupplierAdFlag;
import org.agent.based.tendering.jpa.models.SupplierSpecification;
import org.agent.based.tendering.jpa.models.Tender;
import org.agent.based.tendering.jpa.models.TenderCost;
import org.eclipse.jetty.jmx.MBeanContainer;
import org.eclipse.jetty.server.Server;
import org.eclipse.jetty.webapp.WebAppContext;

public class WebAgent extends AbstractAgent {

    private final Server server;

    public WebAgent() {
        server = new Server(8080);
        queue = new ConcurrentLinkedQueue<>;
    }

    protected void takeDown() {
        try {
            server.stop();
        } catch (Exception e) {
        }
        super.takeDown(); //To change body of generated methods, choose Tools | Templates.
    }

    protected void setup() {
        super.setup(); //To change body of generated methods, choose Tools | Templates.
        addBehaviour(new WebContainerHandler());
        addBehaviour(new RequestsForward(3));
    }

    private class WebContainerHandler extends OneShotBehaviour {
        public void action() {
            new Thread(new Runnable() {
                public void run() {
                    try {
                        MBeanContainer mbContainer = new MBeanContainer(ManagementFactory.getPlatformMBeanServer());
                        server.addBean(mbContainer);
                        org.eclipse.jetty.webapp.Configuration.ClassList classlist =
                                org.eclipse.jetty.webapp.Configuration.ClassList.setServerDefault(server);
                    }
                }
            });
        }
    }
}
classList.addAfter("org.eclipse.jetty.webapp.FragmentConfiguration",
"org.eclipse.jetty.plus.webapp.EnvConfiguration",
"org.eclipse.jetty.plus.webapp.PlusConfiguration");

WebAppContext webapp = new WebAppContext();
webapp.setContextPath("/");
webapp.setWar("web/TenderingUI.war");
server.setHandler(webapp);
org.eclipse.jetty.plus.jndi.Transaction transactionMgr = new

server.setStopAtShutdown(true);
server.start();
server.join();
} catch (Exception ex) {
    log.error(" action -- error starting server ", ex);
doDelete();
}
}
}, "Server Thread").start();

private static ConcurrentLinkedQueue<EW> queue;

private class RequestsForward extends TickerBehaviour {

    public RequestsForward(int seconds) {
        super(WebAgent.this, (long) (seconds) * 1000l);
    }

    protected void onTick() {
        EW pair;
        while ((pair = queue.poll()) != null) {
            Class<? extends ModelTemplate> aClass = pair.entity.getClass();
            if (aClass.equals(ProcurementItem.class)
                || aClass.equals(Specification.class)) {
                ACLMessage acl = new ACLMessage(ACLMessage.INFORM_IF);
                acl.addReceiver(new AID(SPECIFICATION_AGENT,
                        AID.ISLOCALNAME));
                try {
                    acl.setContentObject(pair);
                    send(acl);
                } catch (IOException ex) {
                    Logger.getLogger(WebAgent.class.getName()).log(Level.SEVERE, null,
ex);
                }
            } else if (aClass.equals(EvaluationCriteria.class)) {
                ACLMessage acl = new ACLMessage(ACLMessage.INFORM_IF);
                acl.addReceiver(new AID(EVALUATION_CRITERIA_AGENT,
                        AID.ISLOCALNAME));
                try {
                    acl.setContentObject(pair);
                } catch (IOException ex) {
                    Logger.getLogger(WebAgent.class.getName()).log(Level.SEVERE, null,
ex);
                }
            } else {
                // Handle other classes
            }
        }
    }
}
send(acl);
} catch (IOException ex) {
    Logger.getLogger(WebAgent.class.getName()).log(Level.SEVERE, null, ex);
}
}
} else if (aClass.equals(Tender.class)) {
    ACLMessage acl = new ACLMessage(ACLMessage.INFORM_IF);
    acl.addReceiver(new AID(ADVERTISEMENT_AGENT, AID.ISLOCALNAME));
    try {
        acl.setContentObject(pair);
        send(acl);
    } catch (IOException ex) {
        Logger.getLogger(WebAgent.class.getName()).log(Level.SEVERE, null, ex);
    }
}
} else if (aClass.equals(SupplierSpecification.class) || aClass.equals(SupplierAdFlag.class) || aClass.equals(TenderCost.class)) {
    ACLMessage acl = new ACLMessage(ACLMessage.INFORM_IF);
    acl.addReceiver(new AID(EVALUATION_AGENT, AID.ISLOCALNAME));
    try {
        acl.setContentObject(pair);
        send(acl);
    } catch (IOException ex) {
        Logger.getLogger(WebAgent.class.getName()).log(Level.SEVERE, null, ex);
    }
}
} else if (aClass.equals(Notification.class)) {
    ACLMessage acl = new ACLMessage(ACLMessage.INFORM_IF);
    acl.addReceiver(new AID(NOTIFICATION_AGENT, AID.ISLOCALNAME));
    try {
        acl.setContentObject(pair);
        send(acl);
    } catch (IOException ex) {
        Logger.getLogger(WebAgent.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public static class EW implements Serializable {

    private ModelTemplate entity;
    private int operation;

    public ModelTemplate getEntity() {
        return entity;
    }
}
public int getOperation() {
    return operation;
}

public static void queueRequest(ModelTemplate entity, int operation) {
    if (entity.getId() == null) {
        return;
    }
    EW e = new EW();
    e.entity = entity;
    e.operation = operation;
    queue.add(e);
}