CHAPTER 1

INTRODUCTION

1.1 GENERAL

Agent can be defined as a component that, given a goal could act in the place of a user within its domain knowledge. Agents are also called intelligent agents, as intelligence is a key component of agency. Agent oriented approach can be viewed as next step of Object Oriented approach. The research work attempts to demonstrate the concept of developing Multi-Agent platform for processing of Bio-signals. It also demonstrates the concept of developing agents using JADE – Java Agent DEvelopment framework.

There are four fundamental phases to the software development lifecycle: planning, analysis, design, and implementation. When implementing the software development lifecycle, it is often useful to have some formal guidelines (i.e. A list of steps and deliverables) on how to progress through these phases. This is the task of a methodology Dennis and Wixom (2000). A methodology saves time and effort by crystallizing the important steps that has to be followed, and as a result, providing with the right “direction”. A methodology, thus, essentially acts like a “recipe,” which helps to implement the solution by specifying some of the steps of the process. The importance of a methodology in the software development cycle can, therefore, not be overstated.

Agent-based software engineering is a relatively new field and can be thought of as a evolution of object-oriented programming (Odell (2000).
Though agent technology provides a means to effectively solve problems in certain application areas, where other techniques may be deemed lacking or cumbersome, there is a current lack of mature agent-based software development methodologies. This deficiency has been pointed out as one of the main barriers to the large-scale uptake of agent technology [20]. Thus, the continued development and refinement of methodologies for the development of multi-agent systems is imperative, and consequently, an area of agent technology deserving significant attention.

Current methodologies exist for the development of multi-agent systems including Gaia Wooldridge et al (1998), MESSAGE Caire et al (2002), and Cassiopeia Collinot et al (1996). Some good reviews are provided in Bussmann et al (2004); Luck et al (2004) and Wooldridge (2002). Most current methodologies attempt to adapt object-oriented analysis and design methodologies to agent-based design Wooldridge (2002) and in addition, follow a top-down approach. It has been pointed out that adapting object-oriented analysis and design methodologies to multi-agent system development has several disadvantages Wooldridge (2002) mainly arising from the fact that objects and agents provide different abstractions, and as a result, should be thought at different levels Odell (2000). In addition, the wholly top-down approach assumed by many of the current methodologies is not sufficient for systems containing existing resources which need to be utilized within the multi-agent system.

The proposed methodology for multi-agent system does not attempt to extend object-oriented techniques, instead focusing on agents specifically and the abstractions provided by the agent paradigm. Furthermore, it combines a top-down and bottom-up approach so that both existing system capabilities (including those provided by legacy software and people) and the applications overall needs (based on the requirements) can be accounted for.
As mentioned above, not explicitly accounting for existing systems is a point lacking in many of the currently available methodologies for multi-agent system development. The proposed methodology attempts to formalize the analysis and design phases of the agent-based software development life cycle. The formalization of the planning and implementation phases of the software development life cycle are currently outside the scope of the methodology, though some brief pointers are given.

The design phase specifically focuses on the JADE platform, and the concepts provided by it. JADE is the abbreviation for the Java Agent DEvelopment Framework and has been developed by the Telecom Italia Lab (TILAB) in Italy, in compliance with the FIPA (Foundation for Intelligent Physical Agents) specifications www.fipa.org. FIPA is a non-profit organization geared at producing standards for the interoperation of heterogeneous agents. Essentially, JADE is a middle-ware (written entirely in the Java language, using several Java technologies), which simplifies the implementation of multi-agent systems by providing a set of graphical tools that support the debugging and deployment phases. The agent platform can be distributed across multiple machines, regardless of the underlying operating system, and the configuration controlled via a remote graphical user interface. More information on JADE can be found at http://jade.tilab.com. By specifically focusing on the JADE platform in the design phase, the process can move straight to implementation afterwards, without having to tediously adapt the results of the design phase to an agent platform of choice. This will obviously result in significant time gains for the design phase, in addition to providing with a much clearer picture on how to progress in implementation.
1.2 FEATURES OF JADE

The following is the list of features that JADE offers to the agent programmer:

- Distributed agent platform. The agent platform can be split among several hosts (provided they can be connected via RMI4). Only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as Java threads and live within Agent Containers that provide the runtime support to the agent execution.

- Graphical user interface to manage several agents and agent containers from a remote host.

- Debugging tools to help in developing multi agents applications based on JADE.

- Intra-platform agent mobility, including transfer of both the state and the code (when necessary) of the agent.

- Support to the execution of multiple, parallel and concurrent agent activities via the behaviour model. JADE schedules the agent behaviours in a non-preemptive fashion.

- FIPA-compliant Agent Platform, which includes the AMS (Agent Management System), the DF (Directory Facilitator), and the ACC (Agent Communication Channel). All these three components are automatically activated at the agent platform start-up.
Many FIPA-compliant DFs can be started at run time in order to implement multi-domain applications, where a domain is a logical set of agents, whose services are advertised through a common facilitator. Each DF inherits a GUI and all the standard capabilities defined by FIPA (i.e. Capability of registering, deregistering, modifying and searching for agent descriptions; and capability of federating within a network of DF’s).

Efficient transport of ACL messages inside the same agent platform. Infact, messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When crossing platform boundaries, the message is automatically converted to/from the FIPA compliant syntax, encoding, and transport protocol. This conversion is transparent to the agent implementers that only need to deal with Java objects.

Library of FIPA interaction protocols ready to be used.

Automatic registration and deregistration of agents with the AMS.

FIPA-compliant naming service: at start-up agents obtain their GUID (Globally Unique Identifier) from the platform.

Support for application-defined content languages and ontologies.

InProcess Interface to allow external applications to launch autonomous agents.
It is in the early stages of the software development cycle (i.e. Planning), where it is decided which tool (e.g. which programming paradigm) to use, and an assessment is made on whether an agent-based option (among other options) is the most appropriate solution tool. As mentioned, the proposed methodology does not formally cover planning, and assumes that a decision has been made to use an agent-based solution. However, it should be pointed out that the decision to use an agent-based solution should be well thought out, since it may not always be the best option. To help in making this decision, one should consult the literature, and in particular, applications where agent technology has been successfully applied. Some good sources which provide tips and guidelines on when an agent-based solution should (or should not) be used are Bond and Gasser (1998); Jennings and Wooldridge (1998) and Milgrom (2001), in particular. Furthermore, some potential pitfalls associated with agent-based development, which should be taken into consideration when developing or contemplating such a system, are outlined and discussed in Jennings (1999) and Wooldridge and Jennings (1998). These valuable references will save a lot of time and effort, by preventing the unnecessary implementation of an agent-based solution, or, on the other hand, confirming the appropriateness of an agent-based solution, for a particular case.

The proposed methodology is presented as follows: Section 2 gives an overview of the methodology, while also outlining the assumed definition of an agent (Section 2.1) and the hypothetical example used to illustrate the methodology (Section 2.2). Section 3 explains the Agent Oriented Framework and Section 4 briefs about FIPA specifications. Section 5 outlines the steps in the analysis, while Section 6 outlines the steps in the design. Section 7 gives details on how the methodology could be adapted in creating Multi-agent system with JADE and Section 8 briefs about Deployment and Testing procedure. Finally, in Section 9, some conclusions are presented, and further work discussed.