Analysis and Design of Multi-Agent System for Processing Bio-Signals Using Agent Oriented Programming Approach with JADE

Tholkappia Arasu Govindarajan, Vinoth Kumar M, Rajavignesh R

Abstract--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 paper attempts to demonstrate the analysis part of developing Multi-Agent platform for processing of Bio-signals using JADE – Java Agent DEvelopment framework. FIPA specifies a set of standard interaction protocols, which can be used as standard templates to build agent conversations. For every conversation among agents, JADE distinguishes the Initiator role - the agent starting the conversation and the Responder role - the agent engaging in a conversation after being contacted by some other agent. JADE provides ready-made behaviour classes for both roles in conversations following most FIPA interaction protocols. The technical goal is to develop a multi agent platform for processing of bio-signals aiming at assisting medical practitioners in developing standard examination procedures. The JADE platform is a popular, FIPA-compliant platform for the development of multi-agent systems. As the agents on the JADE environment run on Threads, the response time is very less which helps the medical practitioner to make a quick diagnosis.

Index Terms-- Agents, JADE, FIPA, Bio-Signals

I. INTRODUCTION

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 paper 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.

Agent-based software engineering is a relatively new field and can be thought of as an 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.

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 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. 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.
II. 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).
- 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
- Efficient transport of ACL messages inside the same agent platform. In fact, messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures.
- 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.

III. AGENT BASED BIO-SIGNAL PROCESSING SCENARIO

The paper 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. The agents are trained, intelligent system that is capable of setting up the platform for processing the EEG / ECG / EMG waveforms. The agents themselves communicate with each other in decision making process.

The technical goal is to develop a multi agent platform for processing of bio-signals aiming at assisting medical practitioners in developing standard examination procedures.

If a medical practitioner wants to have an expert opinion about EEG / ECG / EMG of his patient, Generic Agent can be invoked to which he has to specify the SSN (Social Security Number) of the patient, the type of the signal and the corresponding data file. The Generic Agent in turn will search for the Specific Agent – EEG Agent, ECG Agent, EMG Agent based on the signal type on the network and if found, the corresponding information will be passed to the specific agent by the Generic Agent. For example the EMG medical practitioner wishes to have an expert opinion, the EMG Agent with all necessary information, will look for an EMG Expert System (HINT, DARE, CANDID, MIYOSYS-II). Getting the Expert knowledge, the interpretation will be sent back to the Generic Agent through EMG Agent. The expert opinion will be displayed on the user side as well as it will be stored in Database by DB Agent for further references.

IV. ANALYSIS

The analysis phase aims to clarify the problem without any (or minimal) concerns about the solution. In the proposed methodology, the analysis phase is carried out through a number of steps, described in the following sections.

STEP 1: USE CASES

Use cases are an effective way to capture the potential functional requirements of a new system. Each use case presents one or more scenarios that demonstrate how the system should interact with the end user or another system to achieve a specific goal. There are a number of standards for representing use cases. The most popular is the Unified Modeling Language (UML) specification www.uml.org, which defines a graphical notation. Though use cases are used extensively by object-oriented practitioners, their applicability is not restricted to object oriented systems, because they are not object orientated in nature Hampton et al (1997). Hence, it is also possible to apply use cases (without modification) to capture the functional requirements of multi-agent systems.

Based on the description of the multi-agent scenario given in the previous section and after interviewing the potential system users, it is possible to build up a preliminary list of possible scenarios. Accordingly, the use cases are defined, and a use case diagram produced as shown in Figure 1.
STEP 2: INITIAL AGENT TYPES IDENTIFICATION

This step involves identification of the main agent types and subsequent formation of a first draft of the agent diagram. The following rules have been adopted in this step:

- Add one type of agent per user/device.
- Add one type of agent per resource (which includes legacy software).

The agent diagram is one of the main artifacts produced in the analysis phase and is progressively refined from Steps 2 to 5. With reference to Figure 3, the agent diagram includes four types of elements:

1. Agent types: the actual agent types, represented by circles.
2. Humans: people that must interact with the system under development, represented by the UML actor symbol.
3. Resources: external systems that must interact with the system under development, represented by rectangles.
4. Acquaintances: represented by an arrow linking instances of the above elements, specifying that the linked elements will interact in some way while the system is in operation. Note that, at this stage, only acquaintances between agents and resources / humans are shown in the agent diagram (i.e. agent-agent interactions are deferred to a later step).

It should be noted that in the agent diagram, unlike UML use case diagrams, a distinction is made explicitly between humans and external systems. Interacting with a human through a user interface presents additional problems with respect to interacting with an external system.

- **The use of a transducer agent**: The transducer agent serves an interface between a legacy system and the other agents in the system.

- **The insertion of a wrapper**: A code is injected into the legacy resource (i.e. software in this case), provided the legacy resource’s code is available. This inserted code will allow the resource to communicate in agent communication language, thus, converting it into an agent.

- **Rewriting of the code**: It involves rewriting the code to mimic the operation and capabilities of the legacy resource (i.e. software in this case), but with the added ability to communicate in agent communication language.

In the agent diagram produced in this step (see Figure 2), the agents are acting as transducers, i.e. as an interface between the external/legacy systems/people, and the other agents in the system. Transducers are seen, in general, as the most practical and efficient method for accounting for legacy systems and are advocated in the proposed methodology. The reason is that by treating the legacy systems as a black-box, there is no need to tamper with or rewrite code, thus providing a quick means to get that resource functioning as part of the multi-agent system. However, in some cases a wrapper may be more relevant, and to an extreme rewriting, but such considerations should be deferred to the design stage.

STEP 3: RESPONSIBILITIES IDENTIFICATION

In this step, for each identified agent type, an initial list is made of its main responsibilities in an informal and intuitive way. The artifact resulting from this process is the responsibility table.

The following rules have been adopted in this step:

- Derive the initial set of responsibilities from the use cases identified in Step 1.
- Consider the agents where these responsibilities are clearer first and delay the identification of responsibilities for other agents to later steps.

By applying the above rules to the Multi-Agent System Development, the consideration of the Generic agent is initiated and respective responsibilities are identified (listed in Table 1).

Many existing methodologies such as Gaia Woodridge et al (2000) and MESSAGE Caire et al (2002) propose a different approach where atomic roles are initially identified and then possibly merged into agent types. However, this approach is considered less intuitive because in some cases it may become difficult to determine how the atomic roles should be aggregated into agent types, i.e. how many agent types there should be and which type should cover which atomic role(s). The definition of agent types then responsibilities, as in the proposed methodology, removes this ambiguity.
**STEP 4: ACQUAINTANCES IDENTIFICATION**

In this step, the focus is on who needs to interact with whom and the agent diagram is updated by adding proper acquaintance relations connecting agents that need to have one or more interactions (Figure 3). The term acquaintance comes from Gaia Woolridge and Jennings (2000), and is used in the same sense in the proposed system.

An obvious acquaintance relation in the multi-agent system is required between different agents: the user and the provider. Then, since a Generic agent must present the detailed report to its user and this information is stored in the database and made available by the relevant provider agents, there will certainly be an acquaintance relation between the Generic agent and the DB agent. Thus, going one step backward, to Step 3, some new responsibilities can be added to the Generic agent and the DB agent and presented in Table 1.

**STEP 5: AGENT REFINEMENT**

In this step, the set of agent types initially identified in Step 2 are refined by applying a number of considerations. These are related to:

- **Support**: what supporting information agents need to accomplish their responsibilities, and how, when and where is this information generated/stored.
- **Discovery**: how agents linked by an acquaintance relation discover each other.
- **Management and Monitoring**: is the system required to keep track of existing agents, or the starting and stopping of agents on demand.

These above considerations are discussed in more detail in the following sections.

### Support

These considerations are highly dependent on the domain, and hence, it is quite difficult to provide generic indications. Once the Generic Agent is fed with the required details, it in turn searches for the specific agent to pass the required messages. If the specific agent is ready the processing of the signal will be initiated. The refined responsibilities table is presented in Table 1.

### Discovery

In the simplest case, agent discovery can be accomplished by means of proper naming conventions. For example, in the proposed system the processing agents are named after the type of signal it processes – EEG Agent, ECG Agent, and EMG Agent.

Adopting naming conventions is very simple and efficient, but has some limitations:

- Agent names must be globally unique.
- Agents which are going to be involved in an interaction must typically be known in advance. This works well provided that it is known in advance that there is one, and only one, such agent.
- Assuming naming conventions is typically not very extensible.

Naming conventions may lead to additional work when applied to an agent that can appear and disappear dynamically. The reason being that a naming convention does not provide any presence information, and therefore, addressed agents may not be available when an attempt is made to contact them.

Naming conventions cannot be adopted when different users may start their own agents and choose names themselves. In such cases, there is no guarantee that name uniqueness is preserved.

A more sophisticated way to solve the agent discovery problem is the adoption of a yellow pages mechanism. This allows discovery of agents on the basis of their characteristics, e.g. the services they provide. A yellow pages mechanism can be fully distributed across all agents in the system or centralized with a single agent (with a well-known name) responsible for it. Even if this choice, at this point, is a high level design choice, considering that the proposed methodology targets the JADE platform, it is strongly suggested to adopt a centralized approach. This approach completely maps to the directory facilitator agent provided by JADE and thus saves a lot of work in successive phases of the development process.

### Management and Monitoring

Other agent types can be added to address issues such as monitoring agent faults and restoring them, creation of supporting agents that are needed only under certain conditions, or providing presence information. Having refined the set of agent types, the process is to go back to Steps 2, 3, and 4 and iterate until sufficiently detailed descriptions of the agent types, their responsibilities, and acquaintance relations, respectively, are reached. On doing this, with respect to the multi-agent system, the artifacts shown in Figure 3 and Table 1 are obtained.

![Fig. 3 Agent diagram for multi-agent system refined after Step 5](image-url)
Step 1: Agent Splitting/Merging/Renaming. By considering system performance and complexity in relation to the agent deployment diagram produced in analysis, it is determined whether agents should be split, merged or left as is.

Step 2: Interaction Specification. All responsibilities in the responsibility table related to an acquaintance relation with another agent are considered, and the interaction table produced for each agent type.

Step 3: Ad-Hoc Interaction Protocol Definition. In the case that an existing interaction protocol can not be used for an interaction, an ad-hoc interaction protocol is defined using a suitable formalism.

Step 4: Message Templates. The interaction table is updated to specify suitable Message Template objects in behaviours to receive incoming messages.

Step 5: Description to be Registered/Searched (Yellow Pages). The naming conventions and the services registered/searched by agents in the yellow pages catalogue maintained by the JADE directory facilitator are formalized. A class diagram form is used as a representation.

Step 6: Agent-Resource Interactions. Based on the agent diagram produced in analysis, passive and active resources in the system are identified, and it is determined how agents will interact with these resources.

Step 7: Agent-User Interactions. Based on the agent diagram produced in analysis, agent-user interactions are identified and detailed.

Step 8: Internal Agent Behaviors. Based on the responsibility table produced in analysis, the agent responsibilities are mapped to agent behaviors. Different types of responsibilities (including interactions) require different types of agent behaviors have been specified.

Step 9: Defining Ontology. An appropriate ontology for the domain is specified, by making a number of considerations.

Step 10: Content Language Selection. By following some rules, a suitable content language is selected. Iterate Steps 1-10. Move back and forth between analysis and design whenever necessary.

Similar to the analysis, the design phase has been carried out by following a number of logical steps, with a certain degree of overlap.

V. DESIGN

Once the problem has been clarified to a sufficient level of detail, a move is made from the analysis to the design phase, which aims to specify the solution. There is not a strong boundary between these two phases, and while iterating on the analysis or design, one can move between the two (see Figure 2.1). Since from this point on, the proposed methodology focuses on the JADE platform (and hence, the constructs provided by it). Carrying out the design phase allows to reach a level of detail that is sufficient enough to have a relatively straightforward transition to the implementation, with the possibility of a significant amount of code being generated.

Similar to the analysis, the design phase has been carried out by following a number of logical steps, with a certain degree of overlap. The steps in the design have been summarized below:

VI. DEPLOYMENT AND TESTING

After completing the steps of analysis and design, the features of JADE platform have been explored for the implementation of multi-agent system. With reference to the design phase, the following agents have been developed using Java language: Generic Agent, DB Agent, EEG Agent, ECG Agent and EMG Agent. The required behaviors and actions were implemented as per the design guidelines and FIPA recommendations. The required table has been created in Database which is accessible through Java DataBase Connectivity (JDBC).

VII. CONCLUSION AND FUTURE WORK

The JADE platform is a popular, FIPA-compliant platform for the development of multi-agent systems. However, prior to this, no formal work on bio-signal processing had been proposed for the analysis and design of multi-agent systems using the JADE platform. The multi-agent system for processing Bio-signals will help the medical practitioners to have a standard examination procedure. It also helps the medical practitioner to interact with the expert in the field his need in order to make a proper judgment in the diagnosis phase. As the agents on the JADE environment run on Threads, the response time is
very less which helps the medical practitioner to make a quick diagnosis.

As the health care industry is at its peak, the latest developments in the technology can be thought of integrating with it. In this direction, there are several issues remaining for future work, which include:

More emphasis on agent internal structure and mechanisms.

Development of multi-agent system to deploy on mobile network with which the patients can be monitored through wireless media.

Developing and accessing Expert System for specific applications.

Bio-metric based applications can also be developed by means of mobility of agents.

REFERENCE


http://www.msci.memphis.edu/franklin/AgentProg.htm l, Proceedings of Third International workshop on Agent Theories.


17. The FIPA AUML Web Site, see: http://www.auml.org/.


