A methodology serves as a guide for the system design when developing a system. In general, a software development methodology comprises of:

- A process, i.e. a sequence of phases and steps that guide the developer in building the system.
- A set of heuristic rules that support the developer in making relevant choices.
- A number of artifacts, i.e. diagrams, schemas or documents representing in graphical or textual form one or more models of the system.
- A suitable notation to be used in the artifacts.
- A set of patterns that can be applied to solve common situations.
- One or more tools that: automate, as much as possible, the phases and steps specified in the process; force consistency between the models produced; highlight problems arising from incorrect design choices, when possible; generate code and documentation, etc.

The focus of the work is on the process and the artifacts that are produced and illustrating them. A draft notation is also introduced to be used
in constructing these artifacts and, where relevant, some heuristic rules and
design patterns are presented. The described process covers the analysis phase
and the design phase and is shown in Figure 2.1. The analysis phase is general
in nature and independent of the adopted platform. Conversely, the design
phase specifically assumes JADE as the implementation platform and focuses
directly on the classes and concepts provided by JADE. Observing Figure 2.1,
it can be seen that there is strict boundary between the analysis and design
phases. Moreover, the methodology is of an iterative nature, thus allowing to
move back and forth between the analysis and design phases and the steps
therein.

At the end of the design phase, it is able to progress straight to the
implementation, which is where the actual coding occurs. In addition, most of
this phase can probably be carried out by means of a proper tool which
automates the implementation process. However, this point, and likewise,
testing and deployment, are not addressed formally in the current version of
the proposed methodology (as shown in Figure 2.1), leaving them as a topic
for further work 3 (see Further Work, Section 9).

The planning stage, like implementation and testing, is not formally
addressed in the proposed methodology. However, For the sake of the
progress, a question is included (see Figure 2.1), which initially asks whether
to use an agent-based solution or not. If the answer is yes, the analysis phase
will be started, while if the answer is no, seek an alternative solution. As
mentioned in the Introduction, the decision on whether to adopt an agent-
based solution is one which should be made only after consulting the
literature, and, if possible, analyzing their problem with respect to some
guidelines Milgrom (2001). After observing problem domains and cases
where an agent-based solution has been implemented effectively, it has been
decided to go for Agent oriented Programming for setting up Platform for Processing EEG / ECG / EMG Waveforms.

In the methodology adopted, some assumptions have been made. These include:

- The definition of an agent defined in Section 2.2 is assumed.
- The JADE platform is the platform of choice for implementation.
- There are a relatively small number of agents (less than 10).
- The organizational structure of system is static, meaning that non-emergent behaviour at runtime is not expected, and thus, not considered.
- Security is not a concern.

2.1 UNDERLYING PRINCIPLE OF ANY AGENT

When one hears the word “agent”, it is unknowingly related to the field of Artificial Intelligence. But today, the use of this term has not remained limited to AI; it is widely used in fields like manufacturing automations and computer science. In computer science an intelligent agent is a software agent that exhibits some form of artificial intelligence that assists the user and will act on their behalf, in performing repetitive computer-related tasks. Such software agents live in computer operating systems, databases, networks etc. Such a wide use of agents has given rise to diverse definitions of the term agent such as autonomous agents, mobile agents, distributed agents, multi-agents etc, each with its own specifications and understanding.
1. Planning

Defining the objectives

Is an agent-based solution the best alternative?

No

Use another technique

Yes

2. Analysis

1 - Use Cases

2 - Initial Agent Types Identification

3 - Responsibilities Identification

4 - Acquaintances Identification

5 - Agent Refinement

6 - Agent Deployment Information

3. Design

1 - Agent Splitting/Merging/Renaming

2 - Interaction Specification

3 - Ad-Hoc Interaction Protocol Definition

4 - Message Templates

5 - Description to be Registered/Search (Yellow Pages)

6 - Agent-Resource Interactions

7 - Agent-User Interactions

8 - Internal Agent Behaviours

9 - Defining an Ontology

10 - Content Language Selection

4. Implementation & Testing

Unit Testing and Deployment

Figure 2.1 Overview of the methodology
Autonomous agents are claimed to act on their own, taking decisions based on the perceived environment to satisfy the internal goals.

Mobile agents in terms of computer science is software and data which can migrate from one processor to another autonomously and continue to execute on the destination processor.

When several agents act together or interact with each other they are said to form a Multi-agent system.

But the essence of all these definitions is a basic fact which defines any agent. “Each agent is situated in, and is a part on some environment. Each senses its environment and act autonomously upon it. No other entity is required to feed it input, or to interpret and use its output. Each acts in pursuit of its own agenda or pursuing goals designed in by some other agent. Each acts so that its current actions may effect its later sensing, that is its actions effect its environment. Finally, each acts continually over some period of time,” Stan Franklin and Art Graesser (1996).

The key point of an autonomous agent is that it is situated in some environment, change the environment and we might not even have an agent anymore! Say, a robot with only visual sensors when put into an environment with no light is no more an agent because it is no more capable of sensing the environment which makes it impossible to act upon it.

The above discussion of an agent is very general and yields many classes of agents. This can be organized into a hierarchy as shown in Figure 2.2 Stan Franklin and Art Graesser (1996).
2.2 DEFINITION OF AN AGENT

The term agent is very broad and has different meanings to different people. However, on close observation of the literature, it is sufficient to say that two usages of the term agent can be identified: the weak notion of agency and the strong notion of agency Wooldridge and Jennings (1998). The weak notion of agency constitutes the bare minimum that most researches agree on, while the stronger notion of agency is more controversial and a subject of active research.

The weak notion of agency denotes a software-based computer system with the following properties Wooldridge and Jennings (1995):

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

• **Reactivity:** agents perceive their environment and respond in a timely fashion to changes occurring therein.

• **Pro-activeness:** in addition to acting in response to their environment, agents are able to exhibit goal-directed behaviour by taking the initiative.

The strong notion of agency is an extension of the weaker notion, and advocates additional humanistic, mental properties such as belief, desire, and intention Shoham (1993).

Consistent with the weak notion of agency, one author Genesereth et al (1994) has gone so far as to say that software agents are application programs that communicate with each other in an expressive agent communication language. Though at first this definition may seem a little simplistic, it allows one to clearly identify what constitutes a multi-agent system, i.e. agents are just pieces of autonomous code, able to communicate with each other using an agent communication language. The view of agents assumed in the proposed system is based on this definition. Specifically, the system assumes the following definition for an agent:

> agents reside on a platform that, consistent with the presented vision, provides the agents with a proper mechanism to communicate by names, regardless of the complexity and nature of the underlying environment (i.e. operating systems, networks, etc).

Thus, the assumed view is exactly the same as that presented in Genesereth et al (1994); but in addition, the agents have unique names as a
means of identification. This particular view of agents is the only assumption for analysis, while the design is specific to the JADE platform, which is a FIPA-compliant realization of the above vision, i.e. in the design phase, the constructs provided by the JADE platform are assumed.

2.2.1 Shoham’s Agent

Simple understanding of the word AGENT is someone acting on behalf of someone else. Even though the term agent is getting very popular, the term is been used in such diverse ways that it becomes necessary to state its meaning in the current context.

Shoham defines AGENT as follows:

An AGENT is an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices and commitments Shoham (1992). These mental states are precisely defined and have rough correspondence with their real world counterparts. Hence what makes software or hardware component an Agent is a fact that one has chosen to analyze and control it in these mental states. Now the question is not what is agent but “What all entities can be viewed as having these mental states.” Although everything and anything can be described as having mental states, it is not always advantageous to do so.

To ascribe certain beliefs, free will, intension, commitments, ability or wants to a machine or computer program is legitimate when such an ascription expresses the same information about the machine that it expresses about the person.
It is useful when the ascription helps us understand the structure of the machine, its past and future behavior, or how to repair or improve it (Shoham 1992).

Let's elaborate this further with an example of a simple light switch. A light switch can be viewed as an agent who is capable of transmitting current at will. Flicking on the switch is our way to communicate our desire to it. Even though it is legitimate to model a light switch as an agent, it does not give us any information; we already have a very simple mechanical description of working of the switch. On the other hand, it turns out to be useful to model complex components such as robots, people, operating systems, etc., which helps us acquire knowledge about their operations.

When we said that anything can be ascribed with a mental state, it cannot be done so arbitrarily. Some rules must be stated so as to justify ascribing a mental quality of a particular component of the machine.

Below we define some elements which are required for justification of such ascription:

- Precise theory regarding the particular mental category. This theory should have clear semantics and should correspond to the common sense meaning of it. Well, the correspondence will not be exact always. It is up to the consumer of the theory to decide if the correspondence is sufficiently close. For example, in distributed systems, machines communicate with each other to perform certain tasks. Using the mental state semantics it becomes very easy to make statements like “processor B knows that processor
A does not know B is up, hence B will not send any message to A”

- Demonstration that the component of the machine obey the theory. Consider Distributed systems example. It uses knowledge operator which employees’ standard world semantics, but here the possible world is given a strong interpretation. Here possible world means global state of the system. For a machine in this system the world accessible to it is the collection of all global states which are same as its local state.

- Demonstration that the theory plays a nontrivial role in analyzing and understanding of the machine. By using knowledge operator it is easy to prove certain properties of distributed protocols.

2.3 AOP VS.OOP

2.3.1 Object Oriented Approach

1. OOP views the computational system as made up of modules (classes and objects).
2. Each module carries out a particular task.
3. Modules can communicate with each other and have individual ways of handling messages.
2.3.2 **Agent Oriented Approach**

1. AOP specializes the framework by fixing the state (mental state) of the modules (Agents) to consist of components such as beliefs, capabilities, decisions.

2. Various constraints are placed on the mental state of the agent.

3. A computation consists of these agents informing, requesting, offering, accepting, rejecting, completing and assisting one another.

2.3.3 **Difference between Agents and Objects**

Even though both objects and agents communicate with other objects or agents through message passing, in case of objects, message from other object is simply a request to carry out some task. The object to which the request is sent knows how exactly the task is to be carried out. All the tasks which an object can perform are specific, in other words, objects can only perform tasks they are trained for. Agents, on the other hand can affect the behavior of other agent through message passing. An agent A can teach agent B how to carry out some action by first passing the beliefs from B to A, essentially evolving A which can now take actions it previously could not do. For example, say A wants B to open door of room R at time T, if A does not know that there a door to room R then it obviously can not open it. Hence B first has to make A believe that there is a room R with a door, this is does by inserting new believes in A. Once this is done hen B can send A a request to open the door at time T. If it is in the capacity of A to do so, then A will take the action of opening the door to room R at time T.