CHAPTER 5

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 Sections 5.1-5.6.

5.1 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 Section 4.5 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 5.1.
5.2 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).

By applying the above rules to the Multi-Agent System development, the initial diagram shown in Figure 5.2 is obtained.
Figure 5.2 Agent diagram for Multi-Agent System after Step 2

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 5.2, 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 as is highlighted in Steps 6 and 7 of the design phase (Sections 6.6 and 6.7, respectively).

The way external/legacy systems and people that interact with the agents are accounted for in a multi-agent system, is an important consideration (and one that is lacking in many currently available methodologies, as described in the Introduction). One author has defined three techniques (see Figure 5.3) to account for such entities Geneserath et al (1994):

- The use of a transducer agent. The transducer agent serves an interface between a legacy system and the other agents in the system. The transducer agent accepts messages from the agents in the system (in agent communication language), translates them into the legacy systems native language, and forwards these equivalent messages to the legacy system. Similarly, in the reverse direction, the transducer agent receives the legacy systems responses and makes them available to the other agents in the system. In addition to acting as an interface between agents and legacy software, the transducer approach also works for other resources such as files and people (more details given in Section 6.6 and 6.7, respectively).

- 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. This is the most extreme approach, which involves rewriting the code to mimic (and possibly extend) 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. Note that this approach is usually the last resort, when no other options (i.e. the use of a transducer or a wrapper) are deemed practical.

Figure 5.3 Different approaches to account for external/legacy systems

In the agent diagram produced in this step (see Figure 5.2), 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 (though translating between agent communication language and the
resources language is not always trivial). 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 (i.e. in the analysis it suffices to assume the transducer approach).

5.3 **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 Table 5.1 is produced.
**Table 5.1 Responsibility table for multi-agent system after Step 3**

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Agent</td>
<td>Inform the Medical Practitioner that the Agent is ready.</td>
</tr>
<tr>
<td></td>
<td>Let the user enter the SSN, Signal Type and the data source file.</td>
</tr>
<tr>
<td></td>
<td>Communicating Specific Agents with Patient’s info.</td>
</tr>
<tr>
<td></td>
<td>Present the Expert Report with patient’s history.</td>
</tr>
</tbody>
</table>

Many existing methodologies such as Gaia Woodridge et al (2000) and MESSAGE Caire et al (2002) propose a different approach where atomic roles (roughly equivalent to responsibilities defined in this step) 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.

### 5.4 STEP 4: ACQUAINTANCES IDENTIFICATION

In this step, the focus is on who needs to interact with whom and the agent diagram (Figure 5.2) is updated by adding proper acquaintance relations connecting agents that need to have one or more interactions. The term acquaintance comes from Gaia Wooldridge and Jennings (2000), and is used in the same sense in the proposed system.
Figure 5.4 Agent diagram for multi-agent system refined after Step 4

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, Section 5.3), some new responsibilities can be added to the Generic agent and the DB agent and presented in Table 5.2.

Table 5.2 Responsibility table for multi-agent system updated after Step 4

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Agent</td>
<td>Inform the Medical Practitioner that the Agent is ready.</td>
</tr>
<tr>
<td></td>
<td>Let the user enter the SSN, Signal Type and the data source file.</td>
</tr>
<tr>
<td></td>
<td>Identifying the specific agent based on the signal type.</td>
</tr>
<tr>
<td></td>
<td>Communicating Specific Agents with Patient’s info.</td>
</tr>
<tr>
<td></td>
<td>Present the Expert Report with patient’s history.</td>
</tr>
<tr>
<td>DB Agent</td>
<td>Respond to patient’s history retrieval from Generic Agent</td>
</tr>
</tbody>
</table>
5.5  **STEP 5: AGENT REFINEMENT**

In this step, the set of agent types initially identified in Step 2 (see Section 5.2) 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 Sections 5.5.1, 5.5.2, and 5.5.3, respectively.

### 5.5.1  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 5.3.
5.5.2 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.

In the Multi-agent system development, it is assumed that there are some extensibility requirements so that the “one agent per signal type” relation may no longer be true in the future. A yellow pages mechanism is thus adopted for the discovery of provider agents and a yellow pages agent is added to the agent diagram (see Figure 5.1).

For more information about service discovery mechanisms (including centralized and decentralized solutions) in multi-agent systems, can be referred to Campo (2002) and Skylogiannis (2005).

5.5.3 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 (Sections 5.2, 5.3, and 5.4, respectively), 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 5.5 and Table 5.3 are obtained.
Figure 5.5 Agent diagram for multi-agent system refined after Step 5

Table 5.3 Responsibility table for multi-agent system updated after Step 5

<table>
<thead>
<tr>
<th>Agent Type</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic Agent</td>
<td>1. Inform the Medical Practitioner that the Agent is ready.</td>
</tr>
<tr>
<td></td>
<td>2. Let the user enter the SSN, Signal Type and the data source file.</td>
</tr>
<tr>
<td></td>
<td>3. Identifying the specific agent based on the signal type.</td>
</tr>
<tr>
<td></td>
<td>4. Communicating Specific Agents with Patient’s info.</td>
</tr>
<tr>
<td>DB Agent</td>
<td>6. Responding to patient’s history retrieval from Generic Agent.</td>
</tr>
<tr>
<td>EEG Agent</td>
<td>7. Responding to processing request of Generic Agent.</td>
</tr>
<tr>
<td></td>
<td>9. Sending the report to the DB Agent.</td>
</tr>
<tr>
<td>ECG Agent</td>
<td>10. Responding to processing request of Generic Agent.</td>
</tr>
<tr>
<td></td>
<td>12. Sending the report to the DB Agent.</td>
</tr>
<tr>
<td>EMG Agent</td>
<td>13. Responding to processing request of Generic Agent.</td>
</tr>
<tr>
<td></td>
<td>15. Sending the report to the DB Agent.</td>
</tr>
</tbody>
</table>
5.6 STEP 6: AGENT DEPLOYMENT INFORMATION

Another artifact that can be useful to produce is the agent deployment diagram, where the physical hosts/devices agents are going to be deployed are indicated. The agent deployment diagram for the multi-agent system is shown in Figure 5.6. It should be noted that this diagram is not intended to give any detailed information about deployment (in contrast to the UML deployment diagram, where details such as the communication modes between nodes are given). The sole purpose of the agent deployment diagram is to highlight basic deployment requirements that are referred to during design when applying considerations such as agent splitting and merging (Section 6.1) or when considering communication efficiency.

Figure 5.6 Agent deployment diagram for multi-agent system after Step 6

5.7 ANALYSIS SUMMARY

The analysis aims to clarify the problem to a sufficient level of detail, with minimal concern about the solution. The steps in the analysis phase are summarized below:
• Step 1: Use Cases. The system requirements are analyzed and a use case diagram created based on these requirements.

• Step 2: Initial Agent Types Identification. By applying a set of rules, an initial diagram of the multi-agent system called the agent diagram is produced.

• Step 3: Responsibilities Identification. By observing the agent types produced in the agent diagram and applying a set of rules, an initial table of responsibilities is produced, called the responsibility table, for those agents whose responsibilities are clear initially.

• Step 4: Acquaintances Identification. The obvious acquaintances between agents are identified, and subsequently the agent diagram and responsibility table are updated.

• Step 5: Agent Refinement. The agent diagram and responsibility table are updated by applying a number of considerations related to support, discovery, and management and monitoring.

• Step 6: Agent Deployment Information. The agent deployment diagram is produced, where the agents and the physical hosts/devices the agents are going to be deployed are indicated.

• Iterate Steps 1-6.

The important elements gained from carrying out the above steps are the artifacts. These artifacts form the basis for the design phase. The artifacts produced in each step and their relationships are summarized in Figure 5.7
Figure 5.7 Summary of the analysis phase.