Enhancing Legacy Information Systems with Agent Technology

The Case of a Hospital Medical Laboratory

Minh Tuan Nguyen, Dr. Patrik Fuhrer, Prof. Dr. Jacques Pasquier

{minhtuan.nguyen, patrik.fuhrer, jacques.pasquier}@unifr.ch

Abstract

Agent Technology is an emerging and promising technology area of research, which contributes to the development of added-value information systems for complex organizations. The healthcare sector is one of the domains which is attracting researchers in agent technology and developers of agent-based applications.

This paper aims at presenting the advantages of agent technology, particularly software agents, through a case study conducted at the laboratory of the "Hôpital Cantonal de Fribourg". We elaborate on the MediMAS prototype to introduce the concept of personal assistants and to watch them in action. A development methodology for building such an agent-based application is also proposed, with the originality of integrating the system ontology from the very start of the modeling process.

Keywords: Personal Assistant Agent, Multi-Agent Systems, Legacy Information Systems, eHealth-Care.

Table of contents

1. Introduction **3**
1.1. Motivation 3
1.2. What is an Agent? 3
1.3. Agent platforms 4
1.4. Structure of this Document 4
2. A Legacy Laboratory Information System **6**
2.1. Mission and Organization 6
2.2. The cLIS of the HCF 7
2.3. Potential Problems 8
2.4. A Software Agent Solution to cLIS 9
3. The MediMAS Prototype **11**
3.1. Agents as Personal Assistants 11
3.2. Software Agents in Action 12
3.2.1. Environment Setup 12
3.2.2. Notification of Results Availability 13
3.2.3. Acknowledgement of Notification Receipt 14
3.2.4. Problem Detection and Alert 15
3.2.5. Lab Director Intervention 16
4. Development Methodology **18**
4.1. Phase I: Real World System Analysis 18
4.2. Phase II: Domain Ontology Definition 19
4.3. Phase III: Agent-based Modelling 20
4.4. Phase IV: Implementation 22
5. Conclusion **25**
5.1. Achievements 25
5.2. Future Extensions 25
5.2.1. The MediMAS Prototype 25
5.2.2. Methodology Enhancement 25
5.2.3. MediMAS Simulation Tool 25
Appendix A. The business processes of the laboratory of HCF **26**
A.1. Physician accesses WinDMLAB to get the results 26
A.2. Physician asks for results by phone 27
A.3. Laboratory Assistant calls the physician back by phone to give him the result 28
Acknowledgements **29**
References **30**

1. Introduction

1.1. Motivation

The business of today's complex organizations such as hospitals in a healthcare network relies on sophisticated information systems which often inherit many weaknesses from the past. For instance, due to its lack of flexibility, a *legacy information system*[1] cannot integrate the ever-increasing requirements in order to assist the users or to free them from many routine tasks. This weakness of legacy information systems is one of many aspects of the "automation gap".

Another major weakness relates to the increasing physical mobility of users. Many legacy information systems are designed for users working at fixed client workstations in fixed offices. They do not take into account recent advances in mobile technology such as PDAs, mobile phones, smartphones, etc.

In many legacy information systems, the information flow still requires human interaction between actors either face-to-face or through the plain old telephone communication system to get things done (information delivery, alert sending, people search, feedback, etc.).

Automation gap, lack of mobility, and direct human interaction result in an inefficient information flow and data processing:

- Non-automated information search and retrieval are time-consuming for users.
- Error may occur in data transmission by humans (misspelling, data loss, ambiguity, etc.).
- Users must be physically present at either end of the communication link to successfully establish a conversation (i.e. only synchronous interaction).
- Lack of a systematic activity log: the logging of events and actions is often decentralized and not automatic, and thereby makes it difficult to determine the responsibilities of actors when problems or errors occur during a business process.

This research aims at applying software agent technology to overcome these weaknesses. Software agent technology is an emerging and promising technology which attracts great interest in recent years. The design of a software agent layer on top of legacy information systems offers many advantages to users:

- Software agents add interesting properties to information systems: ubiquitousness, intelligence, scalability, systematic error-free management and logging of the information flows, etc.
- An agent-based software layer can help humans to interact efficiently with the information system. Indeed, human effort and time can be saved by transferring routine tasks from humans to software.

1.2. What is an Agent?

The term "agent" appears in a wide spectrum of research areas such as Economics, Physics, Biology, Mathematics, Artificial Intelligence, Software Engineering, etc. Therefore, a unified notion of agent is difficult to extract from the research literature. In this section, we do not aim to coin a new definition, but to highlight the fundamental properties of agents from two published definitions.

> *Definition 1: An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, overtime, in pursuit of its own agenda and so as to effect what it senses in the future [Stan Franklin, 2001].*
>
> *Definition 2: An agent is a small, autonomous or semi-autonomous software program that performs a set of specialized functions to meet a specific set of goals, and then provides its results to a*

[1] A legacy information system represents a massive, long-term business investment in the past [Bisbal, J, 1999], with poor system quality, design and architecture. It is costly to adapt to rapidly changing business requirements.

customer (e.g. human end-user, another program) in a format readily acceptable by that customer [Daniel H. Wagner, 2005].

The first definition proposes the most general notion of agent which may be a person, a robot, a piece of software, etc. The second definition focuses on agents in the software domain which is of interest to us. Both definitions exhibit the following basic properties of software agents:

- *Autonomy:* agents have some degree of control over their actions and can work without intervention of humans.
- *Social ability:* agents can coordinate their actions, and cooperate with other agents to achieve their goals, using a common language to communicate with each other.
- *Reactivity:* agents can perceive their environment and respond to environmental changes.
- *Pro-activeness:* agents can act on their own initiative to achieve their goals instead of simply reacting to the environment.

For our research purposes, we further characterize a software agent as *a running program object, capable to initiate, receive, execute or reject a message autonomously to attain its goals during its life cycle.*

1.3. Agent platforms

An agent platform is a software environment in which agents are incarnated and operate to achieve their goals. The agent platform must provide the following minimum set of functionalities [Bellifemine, F., 2001, 2007]:

- agent management (creating, starting, removing, migrating agents, etc.),
- agent communication,
- supervision of agents, error notification,
- security mechanism.

Today, several platforms have been developed (e.g. JADE [Telecom Italia Group, 2007], JACK [Agent Oriented Software Pty. Ltd., 2006], AgentBuilder [Acronymics, Inc., 2006], Aglet [IBM Research, 2002], etc.) and researches are being conducted to define new platforms for building agent systems.

JADE was selected based on two criteria:

- the selected platform is well-proven,
- it is scalable for our research and experimental purposes.

JADE (Java Agent DEvelopment Framework) is a software framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middleware that complies with the FIPA (Foundation For Intelligent Physical Agents)[2] specifications and through a set of graphical tools that supports the debugging and deployment phases. This 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. JADE has been developed by the Telecom Italia Lab [Telecom Italia Group, 2007], and the Agent and Object Technology Lab at the University of Parma [AOTLab, 2007]. It is open source, cost free and offers the developer complete control over the framework. We refer the interested reader to [Bellifemine, F. L. et al., 2007] for a good introduction to the JADE agent platform.

1.4. Structure of this Document

After this first introductory section, Section 2 presents a case study conducted at the Laboratory of the "Hôpital Cantonal de Fribourg" (HCF)[3]. First, the mission of the HCF Laboratory is defined. Next, its organization, its

[2] FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies [FIPA, 2007].

[3] HCF is the French acronym for Hospital of the province Fribourg, Switzerland [HCF, 2007].

laboratory information system, its problems and weaknesses are examined. Finally, a software agent-based solution to enhance this laboratory information system is proposed.

Section 3 presents the MediMAS (Medical Multi Agent System) prototype from the concept of personal assistant agents through different scenarios that demonstrate the working of software agents.

Next, Section 4 proposes a development methodology which integrates the ontology to build an agent-based system.

Section 5 concludes this paper by summarizing the main achievements of our work and by discussing some extensions and improvements planned for the future.

2. A Legacy Laboratory Information System

2.1. Mission and Organization

A laboratory information system is a computer-based system that supports laboratory functions for collecting, verifying, and reporting test results [Walter, J. D., et al., 1998, p.233]. More precisely, laboratory information systems use computer to:

- analyze input data of specimens,
- store and deliver test results,
- monitor testing quality,
- document laboratory procedures,
- track the status of order processing,
- monitor workflow, and
- assess laboratory productivity.

A laboratory information system reduces the turnaround time between the ordering of a specimen test and the delivery of test results and assists users to interpret the results through reporting capabilities. The quality of services provided to the patients is increased while reducing the healthcare cost.

Legacy laboratory information systems often inherit many features of traditional office environment, for example, the use of telephone network, postal delivery system or facsimile system to exchange information. These technological features may be outdated or difficult to adapt to the growing users needs.

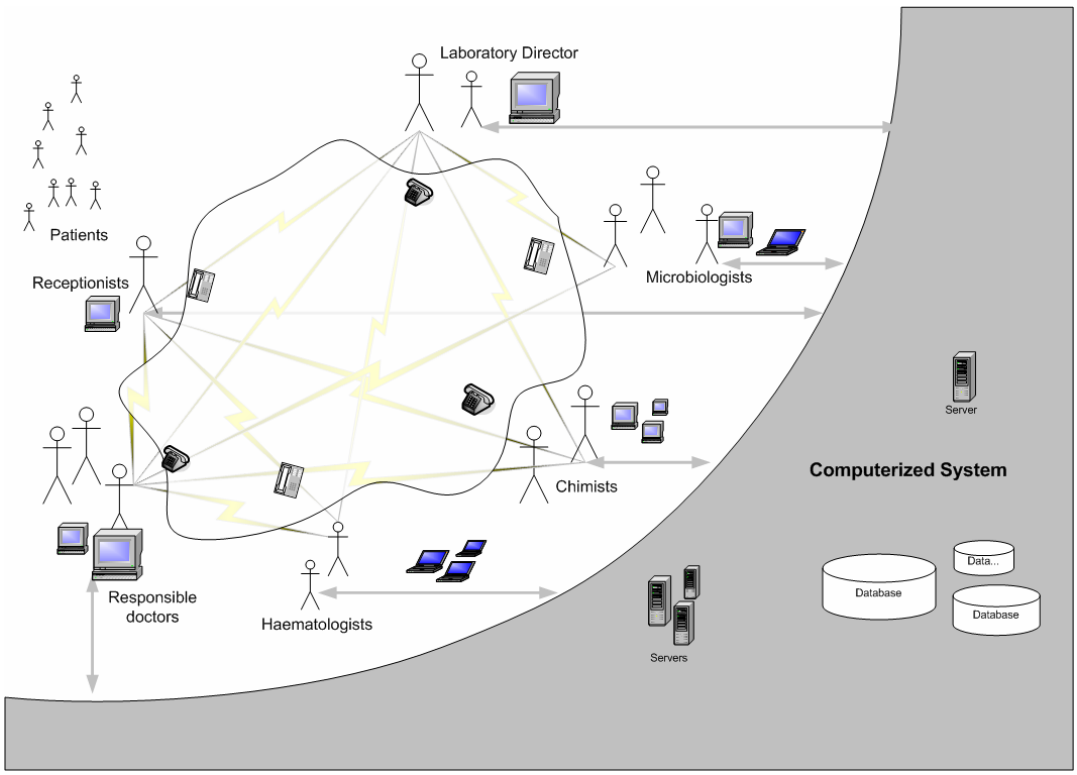

Figure 1. Overview of a Legacy Laboratory Information System

Figure 1 illustrates an overview of a legacy laboratory information system. In this research paper, legacy laboratory information systems are exemplified by the HCF Laboratory in a case study to be discussed later.

The HCF Laboratory[4] provides medical analysis ordered by hospitals in the province. The laboratory contains three departments: haematology, chemistry and microbiology.

The laboratory receives everyday hundreds of orders with specimens, analyzes the specimens, then delivers final results to the requesters (doctors, hospital departments, administrative services, etc).

The method of transmission of test results depends on the urgency level determined by the requester in the analysis order form.

- If the requester does not specify any priority, he will retrieve by himself the lab results whenever he wants to by using the *WinDMLAB Multisite* [Datamed SA, 2007] system.
- The requester can tell the laboratory that he will call to receive the lab results by phone.
- If the requester specifies that his analysis order is urgent, the lab must call the requester as soon as the test results are available.
- If the lab discovers that the test results require immediate medical attention, the lab personnel must call the requester to transmit the critical results, whatever the urgency level in the requester's analysis order form is.

2.2. The cLIS of the HCF

Besides the lab equipment for carrying out medical analysis, the personnel of the Laboratory of the HCF are supported in their daily tasks by *WinDMLAB Multisite*, a laboratory information system developed by Datamed SA, coupled with a traditional telephone communication system (Figure 1). They constitute two major components of the current HCF Laboratory Information System (cLIS).

cLIS ensures the availability of medical results in a centralized database and their transmission:

- between departments of the laboratory,
- between the laboratory and the HCF, and the "Hôpital du Sud Fribourgeois" (HSF)[5]
- between the laboratory and other hospitals in the province of Fribourg.

Each requester (doctors, hospital departments, administrative services, etc.) can access and review the test reports on their patients at any level of detail. It is worth noting that the patients do not have direct access to the system and to their reports.

The *WinDMLAB Multisite* system and the traditional telephone communication system must coexist to achieve all the functionalities as cLIS was initially designed for. Indeed, several scenarios still require the telephone communication system to get things done, for example in these circumstances:

- a lab technologist calls a physician to transmit a patient's test results;
- a physician calls the laboratory to obtain by phone the test results,
- a lab technologist asks, by phone, his director to make a decision in an emergency situation, etc.

Figure 2 illustrates cLIS as a three-layer system in which both the *WinDMLAB Multisite* system and the telephone communication system coexist:

- The first layer defines the information system infrastructure which is composed of servers running different operating systems and application software in a computer network.
- The second layer is the *WinDMLAB Multisite* system.

[4] HCF Laboratory. http://www.hopcantfr.ch/fr/services/medicaux/labo/default.aspx

[5] HSF is the French acronym for another hospital located in the southern part of Fribourg.

- The third layer provides the telephone communication system which allows requesters and laboratory staff to exchange test results via voice and fax.

One can notice that human actors interact with each other directly or indirectly through the second and third layers.

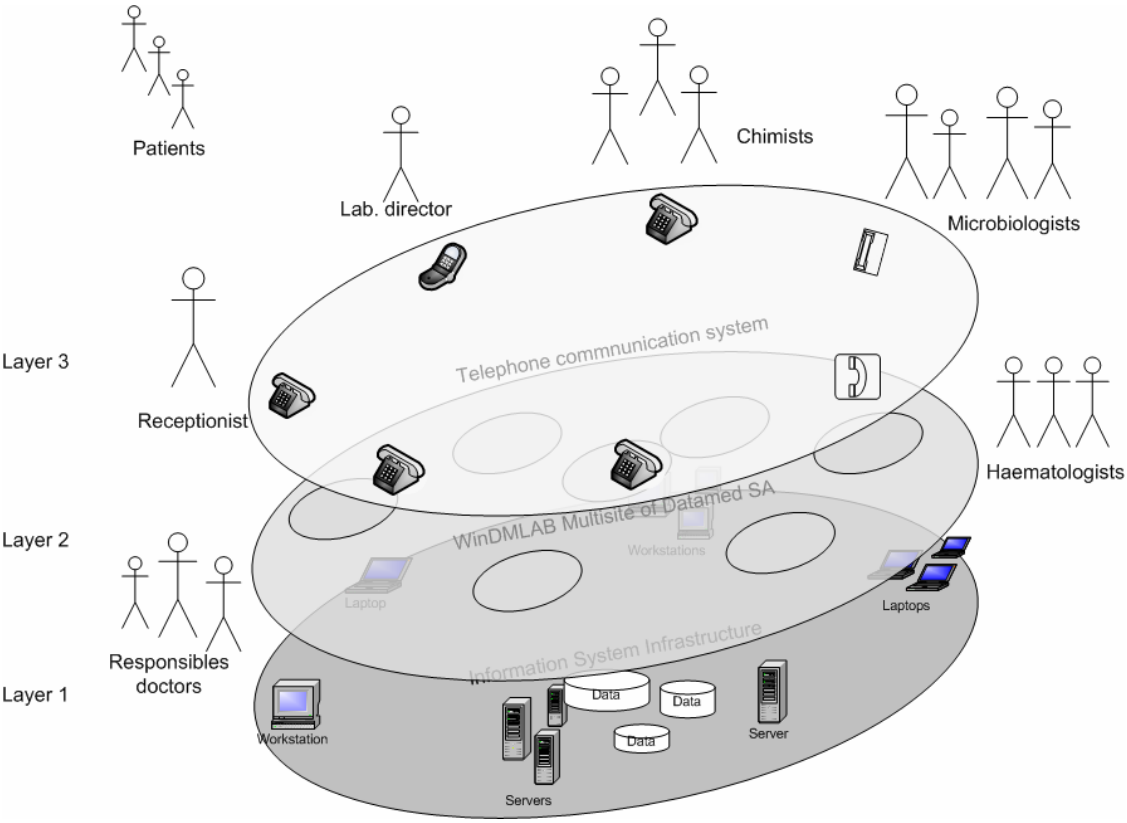

Figure 2. Layers of the current Laboratory Information System

2.3. Potential Problems

cLIS raises numerous potential problems [Ruppen, A., 2007]:

- The quality of services provided by cLIS depends to a more or less extent on human factors. For example, any mistake of a lab technologist in transferring test results to the doctor may cause dramatic consequences on patients.
- cLIS does not allow the requesters to know when results become available.
- The processes which take place in the telephone communication system (layer 3) can not be logged automatically in cLIS for monitoring and tracking purposes.
- Physicians who use cLIS spend a lot of time searching, retrieving, consulting and interchanging the test results.
- To establish a successful phone communication, two actors must be present. Therefore, time is wasted if either one can not reach the other when needed.
- Because of the time-consuming use of cLIS in many scenarios, physicians and laboratory personnel have less time for their real medical activities.

The above identified problems, caused by human operations, often prevent information from flowing smoothly from cLIS to actors and vice versa. These problems illustrate the so called "automation gap" [Gozdan, S. W., 2007; Opalis Software, Inc., 2007]. What is needed is a systematic, strategic approach that automates error-prone human processes.

2.4. A Software Agent Solution to cLIS

The "automation gap" may be filled using different software technologies, for example, JavaSpaces with SMS message technology, Web Services technology, Multi-agent technology, etc.

This section does not aim to compare the advantages and disadvantages of each technology to revamp cLIS. The purpose is to introduce an emerging and promising technology which will allows us to migrate from the legacy *human agent*-centered cLIS toward an enhanced *software agent*-based system.

In cLIS, actors (laboratory personnel, laboratory director, physicians, etc.) are *human agents*. A human agent is a professional characterized by experience, skills, intelligence, reactiveness, proactiveness, and ability to work autonomously and to cooperate with other human agents. They also have weaknesses inherent to human beings.

Our proposal aims at designing *software agents* which will work on behalf of human agents with similar characteristics: intelligence, reactiveness, proactiveness, and ability to work autonomously and to cooperate with other software agents.

In other words, our solution delegates daily routine tasks performed by human agents to software agents.

In this new approach, each actor is assigned a personalized software agent which acts as his *personal assistant*. We also say that the actor is an assistant's owner. When talking about these personal assistants, we could also use the "virtual twin" metaphor [Gachet, A. et al., 2005] or consider them as avatars representing the human like in virtual world.

The assistant receives a list of things to do from its owner, performs the assigned tasks in close cooperation with other software agents, and delivers the final result to the owner.

Figure 3 gives an overview of our solution for cLIS. The software agents are designed on Layer 3, shifting the telephone communication system up to Layer 4.

The software agent solution offers significant advantages for cLIS:

- The features and functionalities of *WinDMLAB Multisite* are maintained, preserving the investment in this legacy laboratory application.
- In the new software agent-based cLIS, the delegation of routine tasks from human to software agents (personal assistants) allows human actors to focus their attention on specimen analysis, test result interpretation, medical decision making, etc.
- The new software agent-based cLIS, coupled with mobile devices (PDAs, mobile phones, smartphones, etc), allows the actors to view the test results transmitted by personal assistants anywhere, anytime.
- All events and actions are systematically logged and centralized to support auditing of the system. Traceability and exception investigation, e.g., to answer a patient's complaint, is also improved.

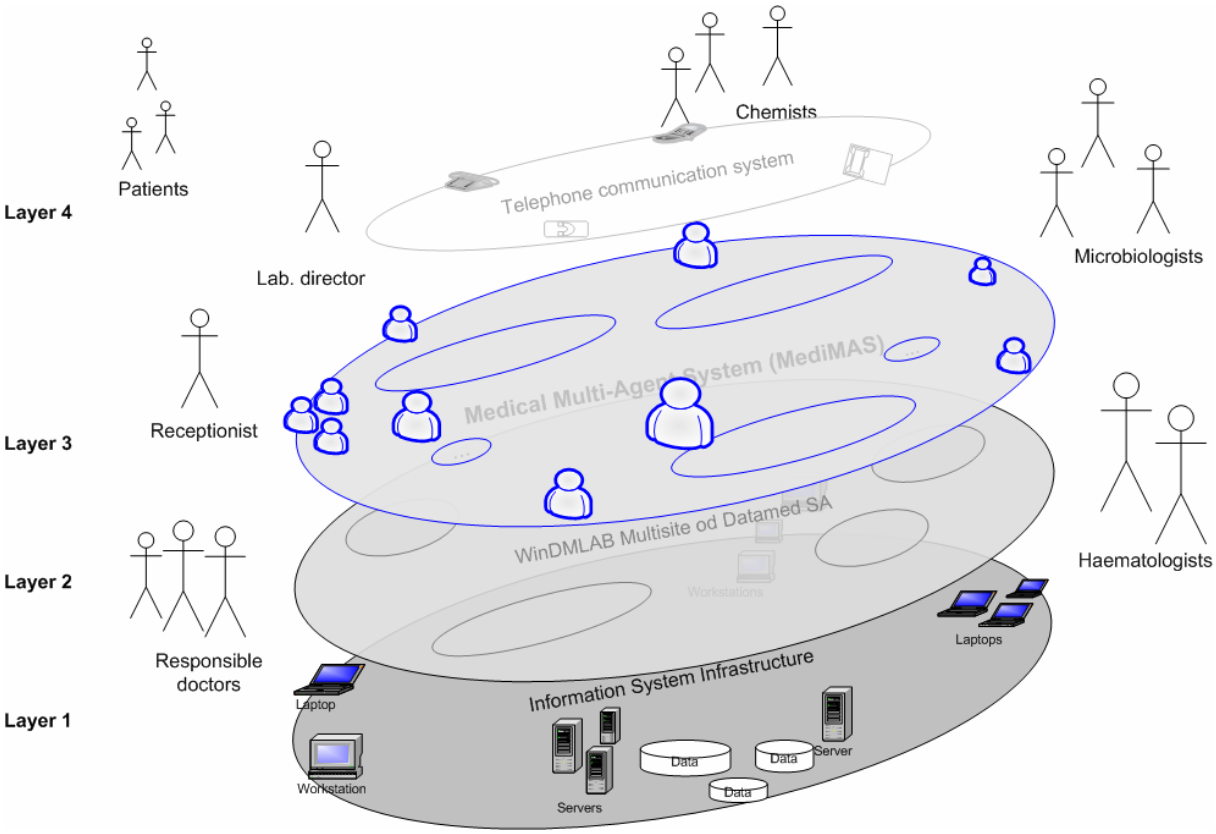

Figure 3. A software agent-based solution

3. The MediMAS Prototype

The MediMAS prototype [Ruppen, A., 2007] is the first experimental implementation of the proposed agent-based solution. A case study was conducted at the HCF laboratory to test the prototype in the real world, and to explore different practical aspects of the solution.

3.1. Agents as Personal Assistants

MediMAS has five agent categories:

- physician agents,
- lab personnel agents,
- lab director agents,
- system supervisor agent, and
- audit agent.

Figure 4 depicts their organization in which the agents assist different categories of humans in their daily tasks. This figure also shows the social ability of agents to cooperate with each other in order to automate the information flow between the actors themselves, as well as between the actors and the cLIS.

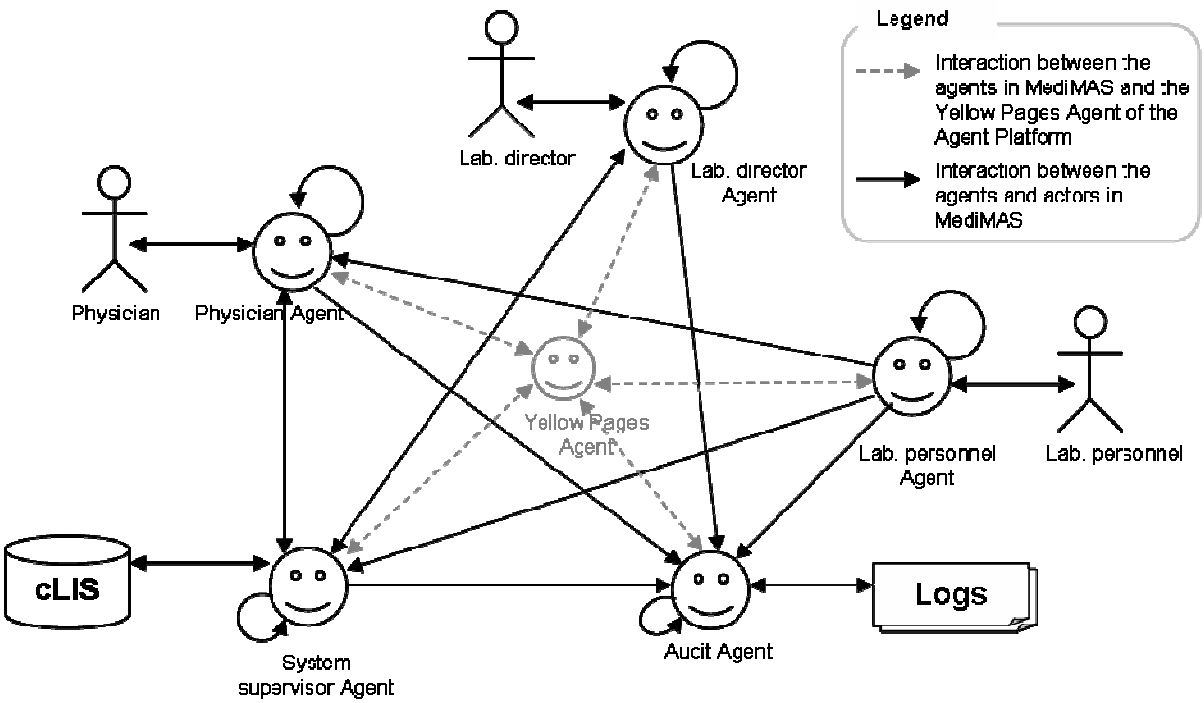

Figure 4. MediMAS Overview

3.2. Software Agents in Action

3.2.1. Environment Setup

In the environment of our MediMAS prototype, the system supervisor agent (ssAgent) plays a central role. Therefore, it is launched first with the JADE platform before starting any other agent. When the setup is complete, the agents attached to the MediMAS's containers[6] (e.g., *MediMAS:MainContainer*, *MediMAS:Container-1*, etc.) appear in the JADE Remote Agent Management window (Figure 5):

- ssAgent is the system supervisor agent.
- adAgent is the audit agent.
- pAgents are the physician agents.
- lpAgents are the lab personnel agents.
- ldAgents are the lab director agents.

One notices that, in the MediMAS system, each human actor (physician, lab personnel, lab director) is assigned an *Agent*, and simultaneously one or more *GuiAgents*. For example, in Figure 5, a single agent *pAgent-TuanAgent* and two GuiAgents, *pAgent-TuanGuiAgent and pAgent-TuanGuiAgent2*, are assigned to the physician *Tuan*.

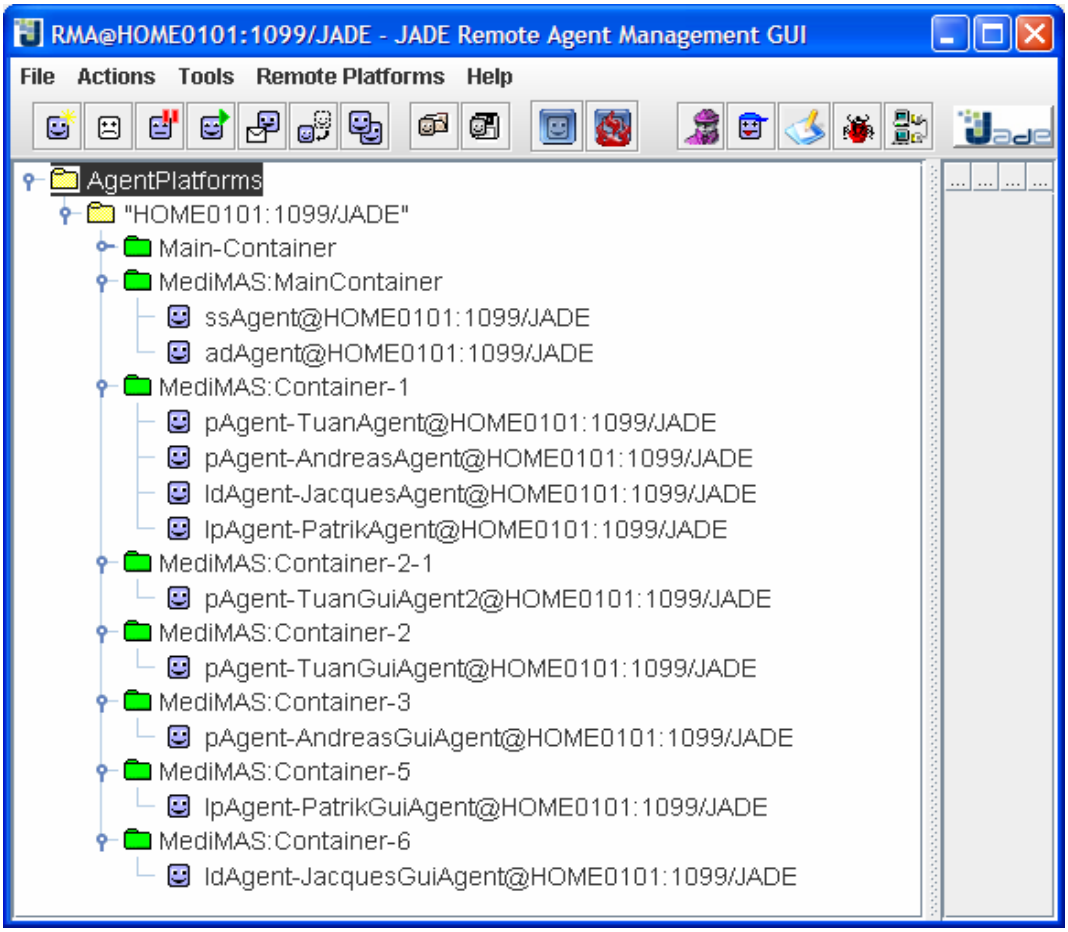

Figure 5. JADE Remote Agent Management GUI

[6] A JADE container is a runtime environment for agents [de Groot, D., Brazier, F., 2006].

We now setup our sample *WinDMLAB database*[7] by feeding it with the fictitious test results of specimens *nlab-007, nlab-008, nlab-009* and *nlab-999* in order to simulate the four test results which are recorded into the database by the lab analysers, and validated by the lab technologist.

Let us introduce the actors who will play different roles in our scenario:

- Patrik is a lab technologist in HCF laboratory. He is working on the following specimens:
  - *nlab-007, nlab-008* and *nlab-009*, ordered by a caregiver Tuan,
  - *nlab-999*, ordered by a caregiver Andreas.
- Tuan is a physician in the HCF and is assigned the ID 3.
- Andreas is a physician in the HSF and is assigned the ID 6.
- Jacques is the lab director.

In the following scenario, starting with the notification of results availability, we study in finer detail the human actors, their assigned personal assistant agents and their interactions.

3.2.2. Notification of Results Availability

- Patrick has finished the analysis of specimens *nlab-007*, *nlab-008*, *nlab-009* and *nlab-999*, ordered by physicians Tuan (ID = 3) and Andreas (ID = 6). The four test results are recorded into *WinDMLAB database*. Table 1 shows the priority of the specimens and their degree of criticality.

Test results degree of criticality	**Priority set by requester in his analysis order form**	
	None	**Urgent**
Non-critical	nlab-007	nlab-008
Critical	nlab-999	nlab-009

Table 1.The four simulated specimens

a) As shown in Table 1, at completion of the *nlab-007* analysis, Patrik observes that the test results are non-critical. In order to notify the availability of the test results to the requester ID = 3, Patrik enters "nlab-007" and "3" in his agent's GUI without checking the checkboxes "Is Critical" and "Is Urgent"[8]. Finally, Patrik clicks the "*Notify Result*" action button to direct his *lpAgent* to announce the availability of test results to the requester (Figure 6).

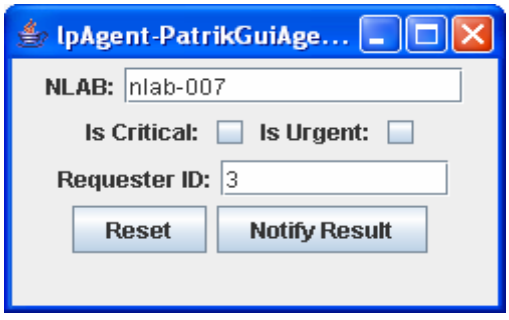

Figure 6. Laboratory Agent's GUI for nlab-007

[7] Our sample *WinDMLAB database* was developed using SQLite RDBMS [SQLite, 2007].

[8] In the future MediMAS enhanced version, Patrik's *lpAgent* will retrieve itself the priority and the degree of criticality from cLIS. Manual input will no longer be needed for the "Requester ID" field, nor does Patrik have to check the "Is Critical" and "Is Urgent" checkboxes. In order to notify a requester, Patrik will only have to input the specimen number, e.g., *nlab-007*.

b) At completion of the *nlab-008* analysis, Patrik observes that the test results are non-critical. Again, he uses his agent's GUI (Figure 6) to notify this result by checking the "Is Urgent" checkbox.

c) At completion of the *nlab-009* analysis, Patrik observes that the test results are critical. Again, he uses his agent's GUI (Figure 6) to notify this result by checking the "Is Urgent" checkbox as well as "Is Urgent" as specified by the requester.

d) In order to notify the availability of *nlab-999* test results to requester ID = 6, Patrik enters "nlab-999" and "6" in his agent's GUI (Figure 7), and checks the "Is Critical" checkbox. Even though the priority was not set by the requester (see Table 1), *lpAgent* automatically checks the "Is Urgent" checkbox. Patrik now clicks the "*Notify Result*" button to direct his *lpAgent* to announce the availability of test results to the requester.

Figure 7. Laboratory Agent's GUI for nlab-999

- Patrik's *lpAgent* sends the announcement of *nlab-007*, *nlab-008* and *nlab-009* to Tuan's *pAgent*, and announces the availability of *nlab-999* to Andreas' *pAgent*.
- It also sends these announcements to *ssAgent* for management purpose.
- *ssAgent* records the announcements and starts to monitor closely the read/unread status of *nlab-007*, *nlab-008*, *nlab-009* and *nlab-999* test results.

3.2.3. Acknowledgement of Notification Receipt

- Concurrently with *ssAgent*, Tuan's *pAgent* receives the announcements and refreshes the list of pending results in the upper pane of its window by adding the new announcements of *nlab-007*, *nlab-008* and *nlab-009* test results, flagged as "available" in the *Status of Announced Result* column (Figure 8).

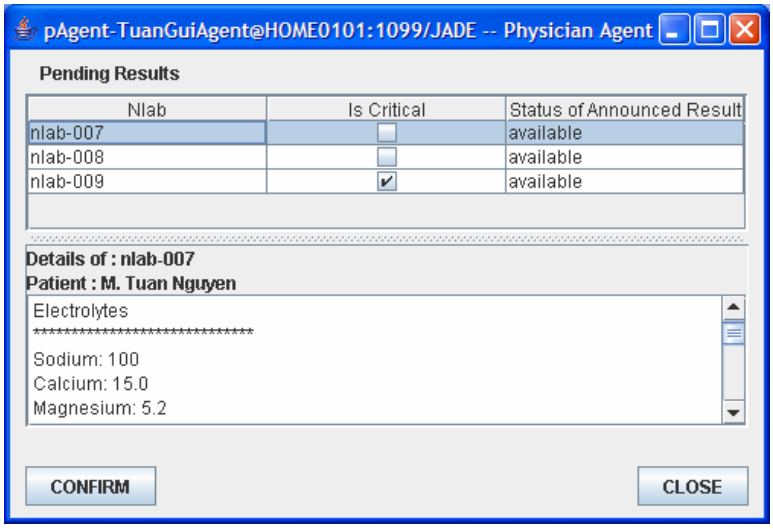

Figure 8. Tuan's Physician Agent GUI

- Tuan clicks on the received announcement *nlab-007* in the list of pending results in order to review the details of the test results.
- Tuan's *pAgent* requests *ssAgent* to retrieve the contents of the *nlab-007* test results.
- Tuan's *pAgent* displays the contents of the *nlab-007* test results in the lower pane of its window.
- Tuan clicks the "confirm" button to acknowledge receipt of the notified announcement of *nlab-007*.
- Tuan's *pAgent* sends the acknowledgement to *ssAgent*.
- ssAgent updates the status of *nlab-007* as "read" in the *WinDMLAB database*, and removes the *nlab-007* announcement from his own internal list. This terminates the monitoring of *nlab-007* by *ssAgent*.
- Once the announcement is flagged as "read", Tuan's *pAgent* removes *nlab-007* from the list of pending results (Figure 9).

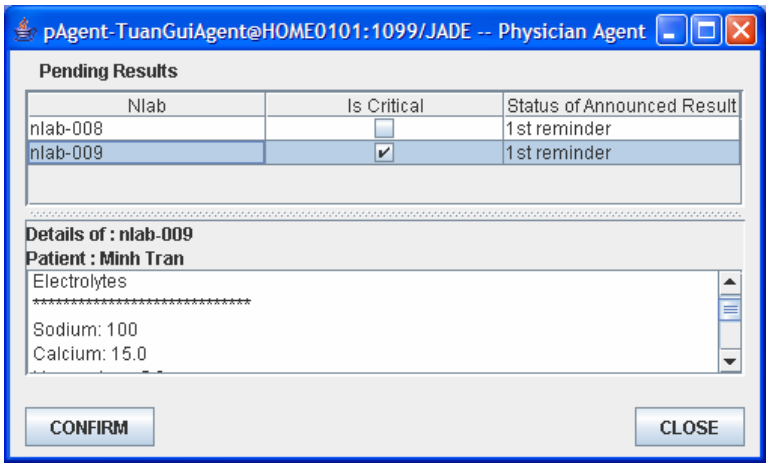

Figure 9. Tuan's Physician Agent GUI - after nlab-007 has been confirmed

One notices that, in the *pAgent's* window, each announcement is first flagged as "*available*" during a predefined time interval, for example, 20 minutes for normal test results; and 10 minute for critical ones. Thanks to the close monitoring of pending announcements, *ssAgent* alerts *pAgent* as soon as an announcement has not been confirmed within the predefined time interval. *pAgent* immediately flags the alerted announcement as "*1st reminder*", then "*2nd reminder* ", and so on in the *Status of Announced Result* column.

- Tuan acknowledges the remaining *nlab-008* and *nlab-009*.

3.2.4. Problem Detection and Alert

For the *nlab-999*, *ssAgent* has not yet received an acknowledgement message from Andreas' *pAgent* within the preset time interval. After three unsuccessful warnings, *ssAgent* escalates up the organizational hierarchy by sending an alert to Jacques' *ldAgent*. Figure 10 illustrates the unconfirmed pending result as seen by Andreas after four reminders.

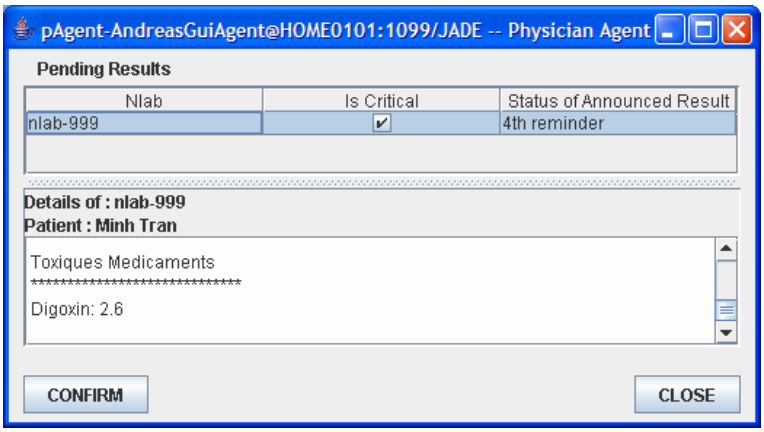

Figure 10. Andreas' Physician Agent GUI

3.2.5. Lab Director Intervention

- Jacques' *ldAgent* receives the *nlab-999* alert from *ssAgent* and displays it in the *ldAgent's* window.
- Jacques clicks on the *nlab-999* alert in order to review it.
- Jacques's *ldAgent* requests *ssAgent* to retrieve the contents of the *nlab-999* test results.
- Jacques's *ldAgent* displays the contents of the *nlab-999* test results in the lower pane of its window.
- Jacques contacts Andreas to manually transmit the test results to him.
- Jacques clicks the "confirm" button to acknowledge receipt of the *nlab-999* alert.
- Jacques's *ldAgent* sends the acknowledgement to *ssAgent*.
- *ssAgent* updates the status of *nlab-999* as "read", and removes the *nlab-999* announcement from his own internal list. This terminates the monitoring of *nlab-999* by *ssAgent*.
- Once the announcement is flagged as "read" in the *WinDMLAB database*, Jacques's *ldAgent* remove *nlab-999* from their respective windows (Figure 11).

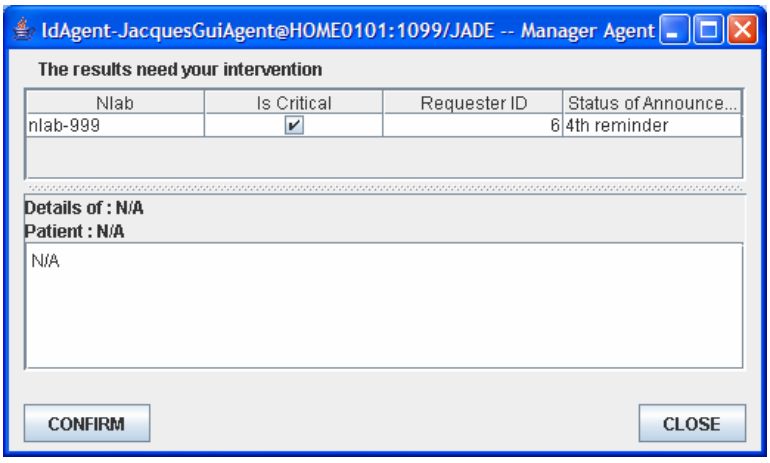

Figure 11. Jacques' Lab Director Agent GUI

Throughout the above simulated scenario, each agent sends to the audit agent (adAgent) the start and stop times of every performed task along with its relevant information (date and time, involved actors, action, etc.). Figure 12 displays an excerpt of the adAgent's log file.

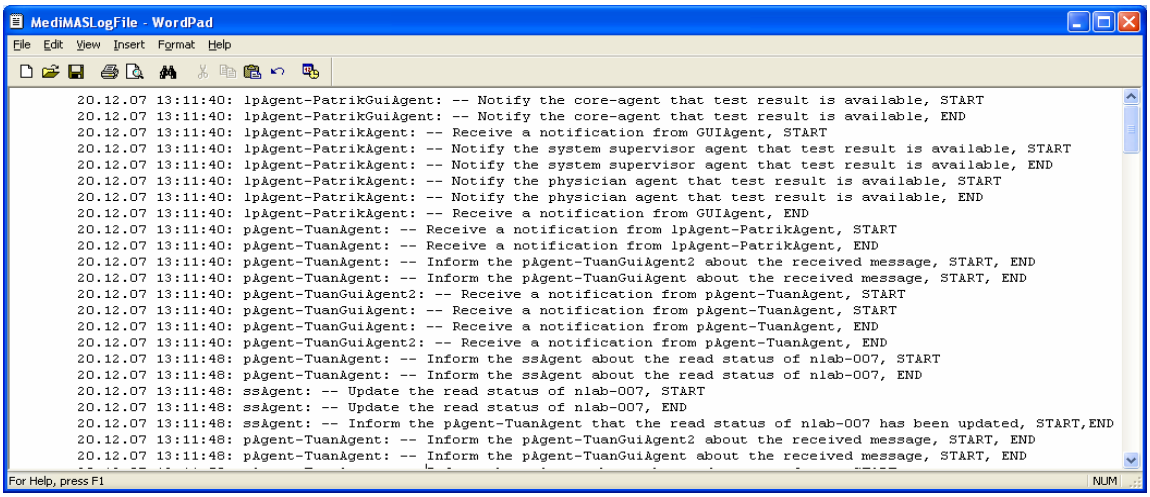

Figure 12. The log file of auditAgent

We have simulated four specimens *nlab-007*, *nlab-008*, *nlab-009* and *nlab-999* to demonstrate the working of assistant agents in the MediMAS prototype and the benefits of a software agent approach to enhance a legacy information system. In the real world where hundreds of specimen analysis are ordered everyday, the software agents in the MediMAS system are valuable personal assistants that increase the productivity of the human actors in the laboratory (the lab personnel, the lab director and the physicians).

4. Development Methodology

The concept of agents was first introduced in the 1970s. However, the development of agent-based systems is a relatively new domain of software engineering. Today, several agent-oriented methodologies have been developed: Gaia [Zambonelli, F. et al., 2005], MaSE [Deloach, A et al., 2005], and MAS-CommonKADS [Iglesias, A. C. et al., 2005], just to name a few. They are based on different theoretical foundations [Henderson-Sellers, B., 2006]: Artificial Intelligence (AI), Object-Oriented (OO), combination of AI and OO, and i* organization modeling framework (Tropos) [Giorgini, P. et al., 2005].

These methodologies contribute significantly to the rigorous and systematic development of agent-based systems. However, most of them do not emphasize the ontology approach which we consider as a new way to share the knowledge between the actors from different categories (users, developers, machines) through the successive phases of the development process.

The JADE_Methodology [Nikraz, M. et al., 2006] is a new agent-oriented methodology that supports the ontology approach. It encompasses the analysis and design phases to develop software agents on JADE platform. However, this methodology proposes to build the ontology at the end of the design phase in order to share the knowledge between software agents. This would not allow us to take full advantage of using ontology in the earlier phases of the development process.

Therefore, we have designed a new "in-house" methodology, inspired by the above-mentioned theoretical foundations, which also integrates the ontology approach. Our methodology has been applied to develop the MediMAS prototype. Figure 13 shows the sequence of the four phases of the methodology, while Figure 17 and the next sections describe them and their relationships.

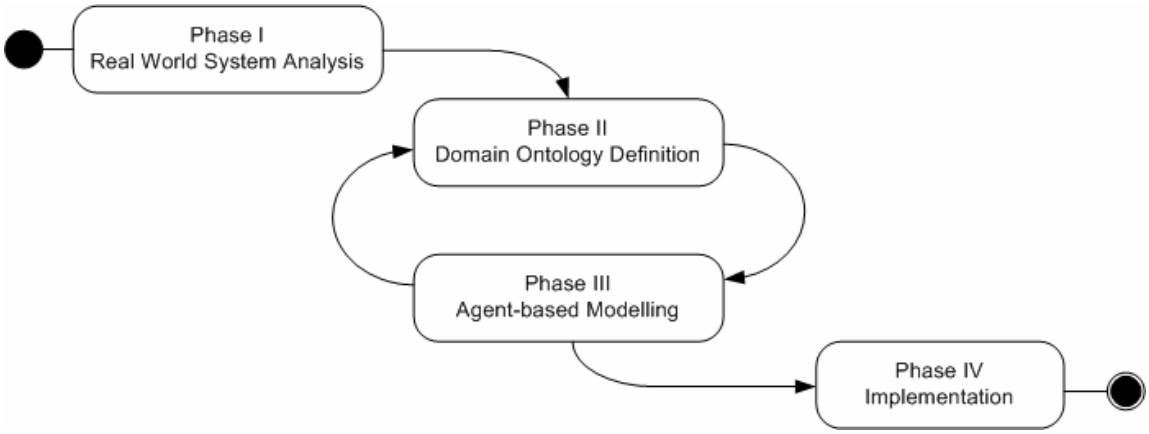

Figure 13. The phases of the methodology

4.1. Phase I: Real World System Analysis

The analyst perceives the current system in order to understand its goals, problems, and its future requirements. This phase aims at defining a common vocabulary and describing the current organization of entities (actors, human agents), use cases, and/or business processes of the system.

The deliverables of Phase I consist in a well-defined set of goals and requirements, the common vocabulary describing the entities with their organization, a set of identified use cases and business processes.

In our case study, the outputs of our real world system analysis are the three-layer information system structure of the HCF Laboratory (Figures 1 and 2), and the UML activity diagrams of its business processes (Figure 14).

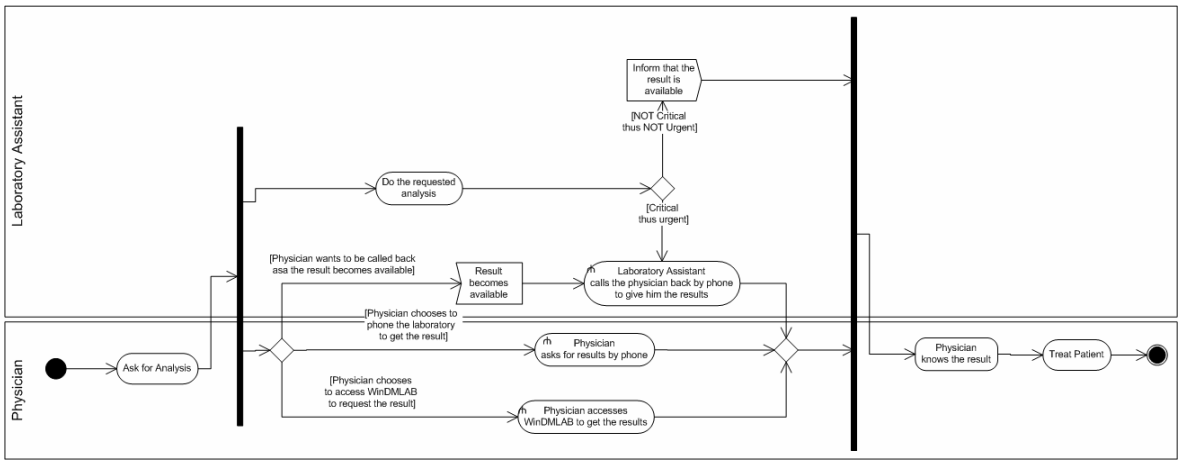

Figure 14. The business processes of the laboratory of HCF

4.2. Phase II: Domain Ontology Definition

The Domain Ontology Definition phase takes the deliverables of Phase I as input and aims at defining the domain or application terminology standards and semantics. To this end, the analyst focuses on concepts, actions, predicates and relations between concepts. In MediMAS, we adopt the following guidelines to build the ontology shown in Figure 15.

- Concepts are substantives (e.g., doctor, patient, analysis, etc.).
- Actions are verbs or verbal phrases (e.g., SendResult, Alert, SendAvailableList, etc.).
- Predicates are expressions that make statements about something, which can be evaluated (true, false and indeterminate), e.g., isTestResultCritical, isResultComfirmed, etc.
- Relations are expressions that establish the relationship between concepts.

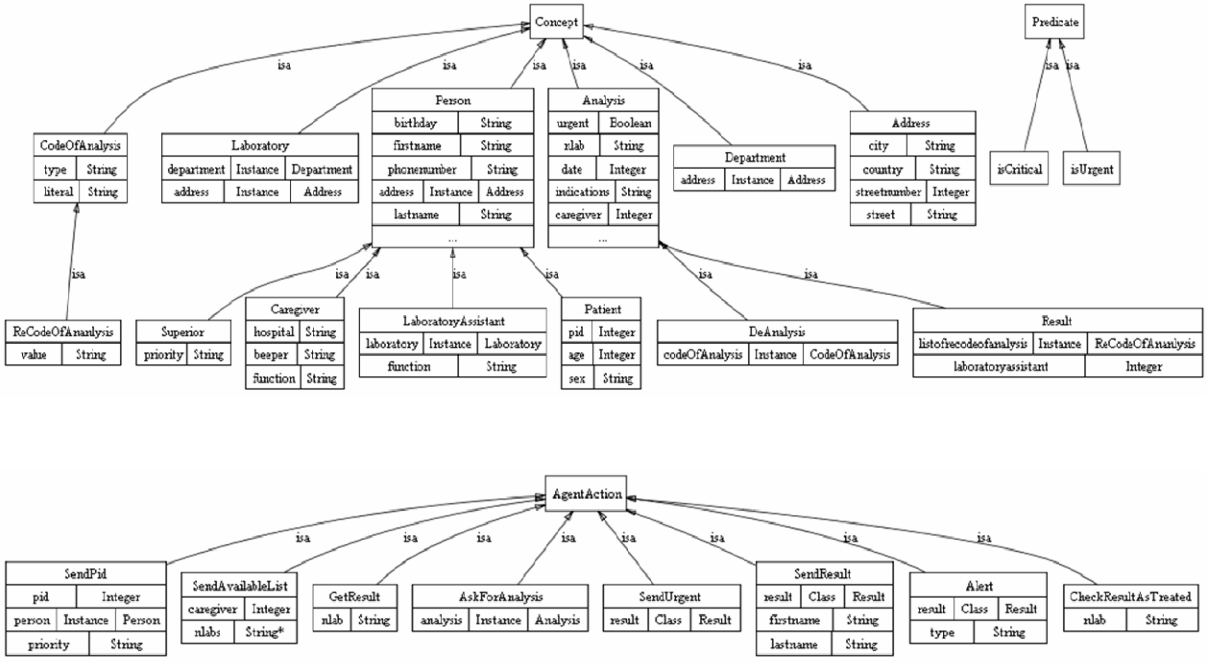

Figure 15. MediMAS's ontology using Protégé suite of tools

The output of this phase is the domain or application ontology, that actors will use to understand each other in their communications.

In software engineering, ontology development tools, such as Protégé [Stanford Center for Biomedical Informatics Research, 2007], TopBraidComposer [TopQuadrant, Inc., 2007], etc., have been developed in order to assist the ontologists to build the domain or application ontology efficiently. Our MediMAS ontology has been developed using the Protégé suite of tools.

4.3. Phase III: Agent-based Modelling

The modelling phase consists in the following set of tasks using the deliverables of Phase I and Phase II as inputs:

- identify and create eligible software agents which will be assigned to actors;
- determine the tasks (also called the responsibilities) of each agent;
- specify the workflow of elementary operations in each task and the agent's operational behavior;
- assign tasks, workflows and behaviors to agents according to their roles in the organization.

Figure 13 and 17 draw our attention on the iterative nature of the tasks within Phase III on one side, and between Phases II and III on the other side. Indeed, successive refinement steps are required in order to enrich the domain ontology as new concepts, actions, predicates, and relations between concepts are identified.

The deliverables of this phase are the documents:

- describing the agents in different categories, and
- specifying all the tasks, workflows and behaviours, and their assignment to agents.

The agent categories and their assigned tasks in MediMAS are summarized in Table 2.

Agent categories	Tasks
Physician Agents	• Receive notification of test results availability from the lab personnel agents. • Receive alerts of unread available test results from the system supervisor agent. • Notify the physician that test results are available • Query the system supervisor agent for test results according to search criteria determined by the physician (N.B.: a single incarnated physician agent is assigned to one physician, called the owner; conversely, one physician can own one or more physician agents). • Receive test results data from the system supervisor agent. • Display test results data for the physician. • Inform the system supervisor agent about the read/unread status of the test results sent to the physician. • Inform the audit agent before and after each action.
Lab personnel Agents	• Notify the system supervisor agent that test results are available. • Notify the physician agents that test results are available. • Inform the audit agent before and after each action.
Lab director Agents	• Receive alerts from the system supervisor agent signalling the abnormal unread status of a given test results. • Report alert to the lab director. • Acknowledge the system supervisor agent that the lab director read the alert sent to him. • Inform the audit agent before and after each action.
System Supervisor Agent	• Extract test results from cLIS, based on the query of the physician agent. • Deliver extracted test results to the physician agent. • Alert the lab director agent as soon as the unread status of a given test result is detected within a predefined check time interval. • Receive test results from the lab personnel agent. • Receive from the physician agent the status "*test results have been read by physician*". • Receive from the lab director agent the status "alert message has been acknowledged by the lab director". • Inform the audit agent before and after every action.
Audit Agent	• Receive the actual action start/end notifications and log them with their date and time.

Table 2. Tasks performed by agent categories

4.4. Phase IV: Implementation

The previous phases I, II and III are platform-independent. In phase IV, the selection of a platform closely impacts the implementation process. In our case study, the JADE platform was selected to implement our MediMAS prototype.

This phase involves the programmer team to implement and test the agent-based system according to the model specifications. To this end, the programmers use the deliverables of the previous phases as inputs, and then translate them into system components which are extensions of the existing classes in JADE, namely:

- designed agents are translated into classes of agents according to terminology used in JADE;
- designed tasks, workflows and behaviours are converted into classes of behaviours in the sense of JADE.

The domain ontology must also be implemented as extensions of the existing ontology in JADE. This task is achieved:

- either by manual coding of vocabulary, bean classes, ConceptSchema, AgentActionSchema, PredicateSchema, etc.,
- or through the bean generator plug-in for Protégé [van Aart, C.J., 2007].

The completion of phase IV results in a multi-agent system that fulfils the defined user goals and requirements and operates on the selected platform.

Figure 16 shows the class diagram of MediMAS as implemented on JADE platform, while Figure 17 provides a graphical summary of the whole methodology.

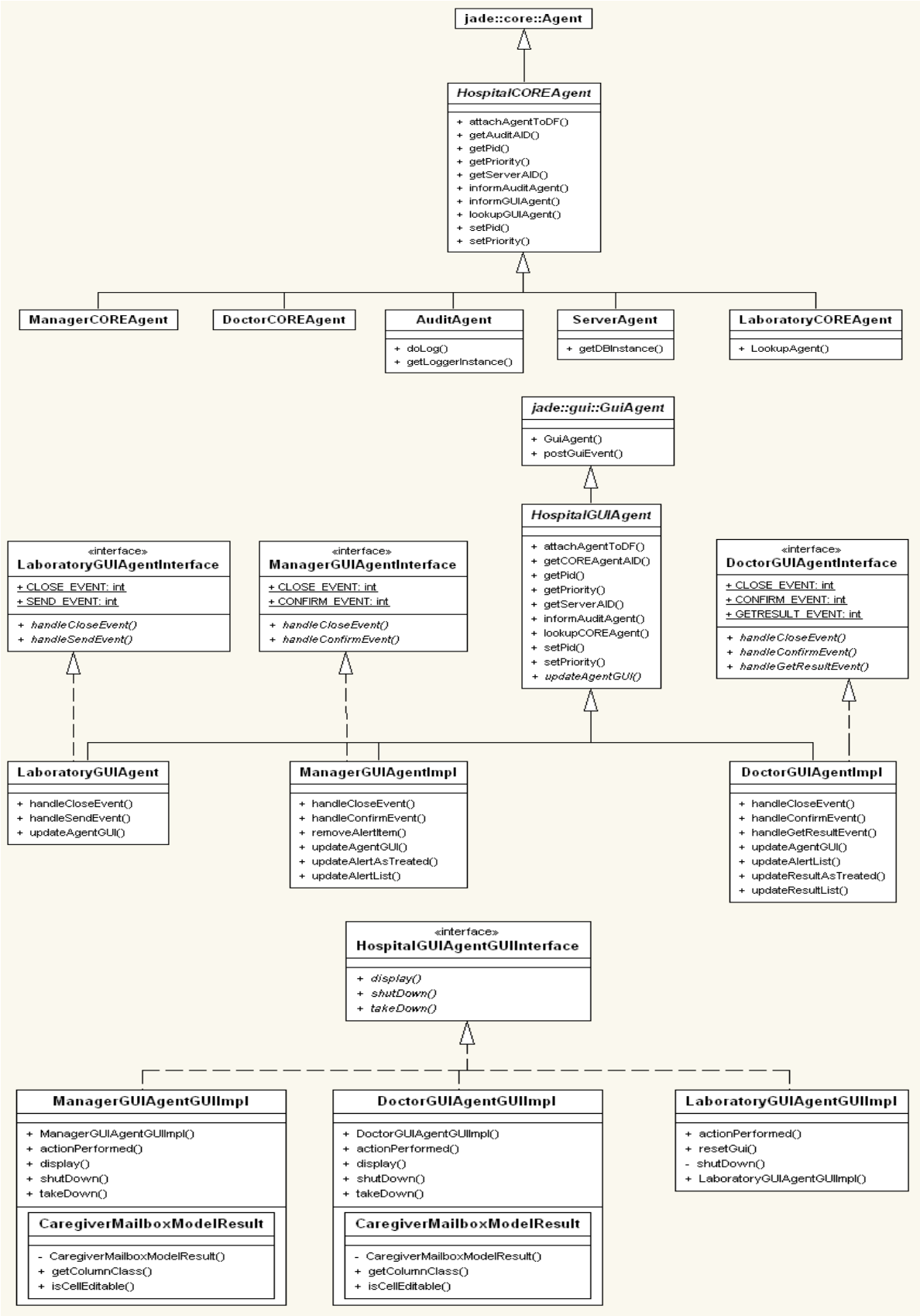

Figure 16. Class diagram of the MediMAS prototype

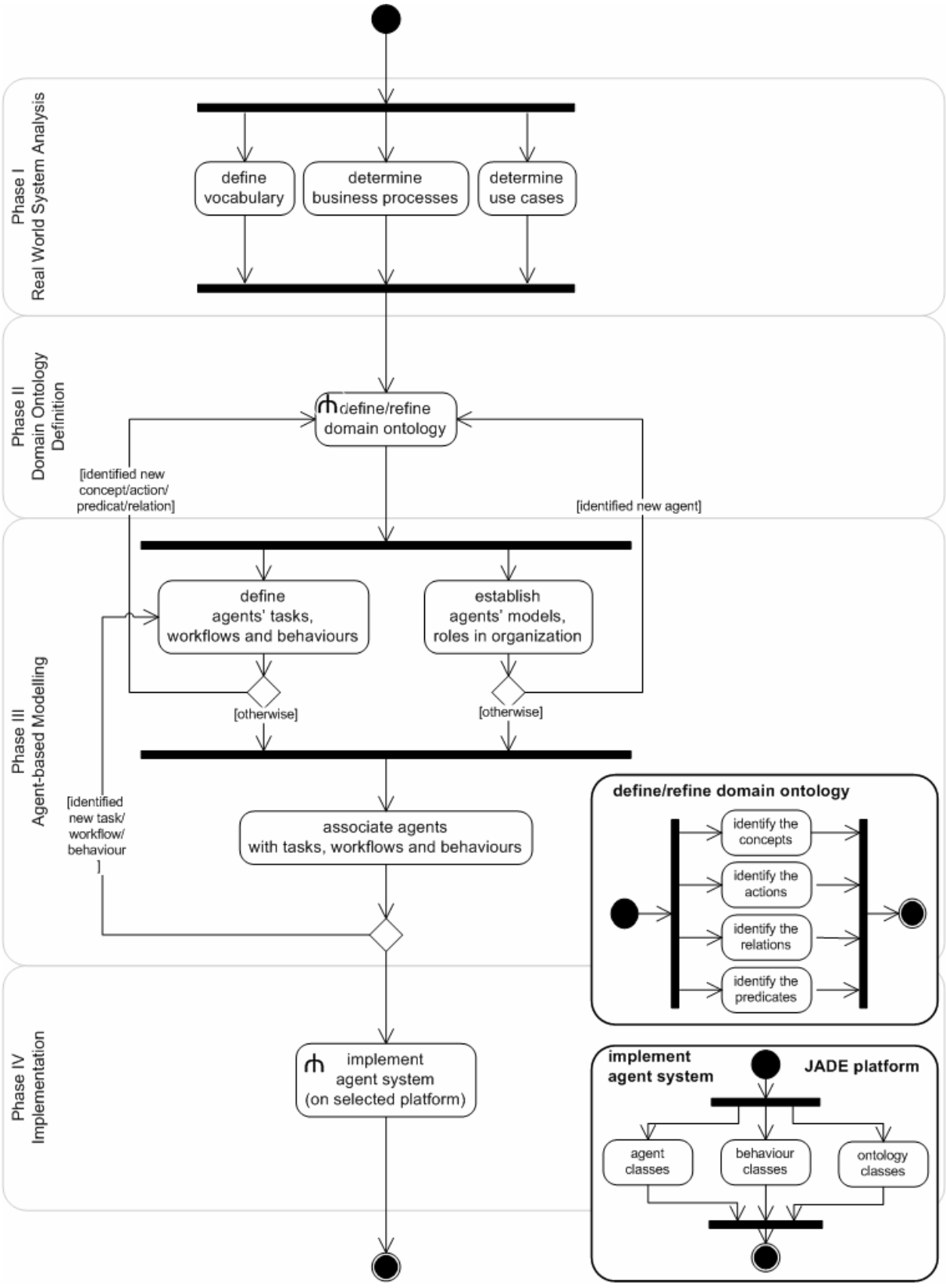

Figure 17. Development Methodology

5. Conclusion

This research paper discusses major features and benefits of the agent-based approach to enhance a Legacy Information System.

The agent-based approach preserves the investment in a Legacy Information System and allows developers to add new features, which aim at filling the automation gap, satisfying the needs of growing user mobility, and providing intelligent assistance to users.

The proposed four-layer design to implement the MediMAS prototype demonstrates the seamless extension of a legacy information system to a multiagent system.

5.1. Achievements

The current version of the MediMAS[9] prototype provides physicians, lab personnel, and lab director with software agents running on desktop computers. These agents act as personal assistants to free the actors from tedious and routine work so that they can really concentrate on their medical activities.

5.2. Future Extensions

5.2.1. The MediMAS Prototype

Our research will extend the model to allow software agents to run on mobile devices (e.g., PDAs, mobile phones, smartphones, etc.). The agents that work for the same owner on different devices must collaborate and synchronize their tasks to efficiently assist the owner who may work anywhere, anytime.

5.2.2. Methodology Enhancement

The light in-house agent-based system design methodology has been defined, and applied in the MediMAS experimental project in HealthCare domain. Future extensions will enhance the methodology with additional modelling possibilities to design more complex real world systems.

5.2.3. MediMAS Simulation Tool

The development of a simulation tool for MediMAS is another topic of our research. The tool offers the HealthCare experts the opportunity to visualize the working of MediMAS prototype by simulation, and to get an insight in the properties of an agent-based system in the HealthCare domain (ubiquitousness, intelligence, reactiveness, proactiveness, scalability, etc).

[9] MediMAS - Version 2.00 (http://diuf.unifr.ch/people/nguyenmi/download/MediMASV2.00.rar)

Appendix A. The business processes of the laboratory of HCF

A.1. Physician accesses WinDMLAB to get the results

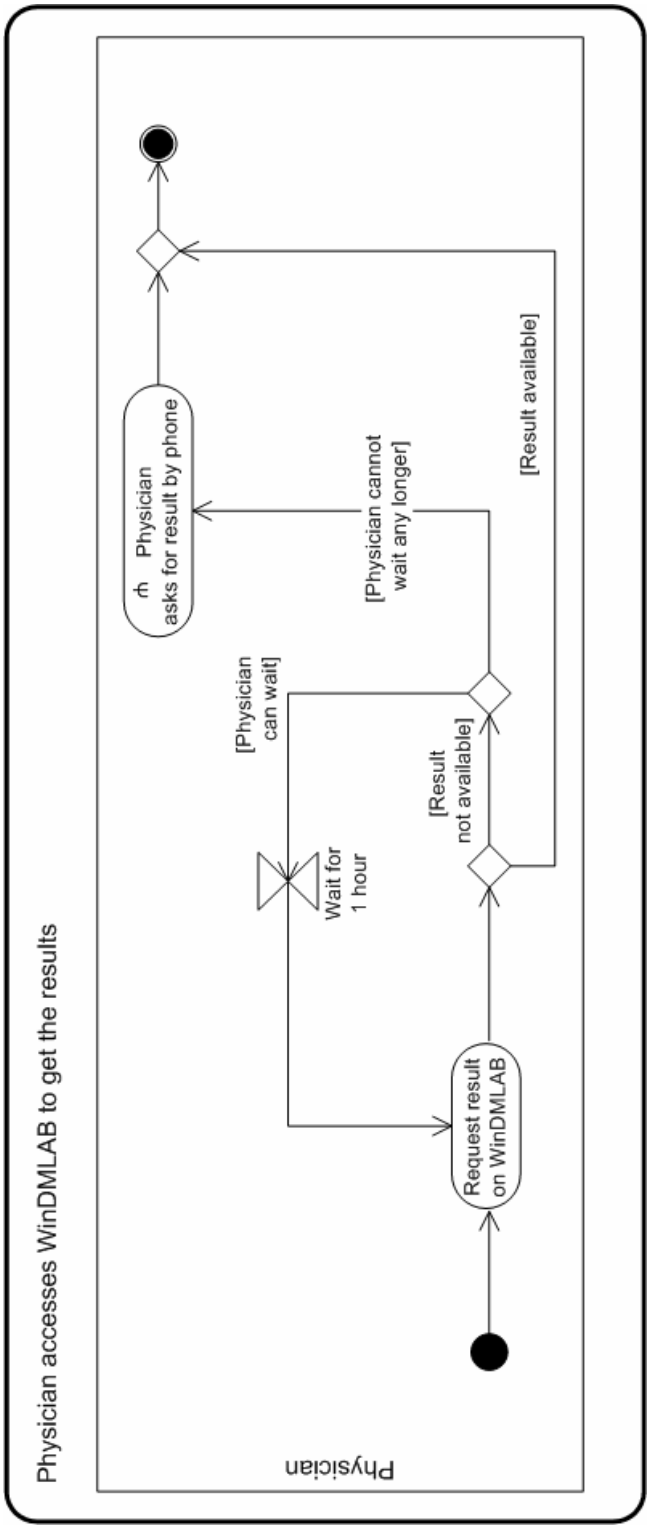

Figure 18. Physician accesses WinDMLAB to get the results

A.2. Physician asks for results by phone

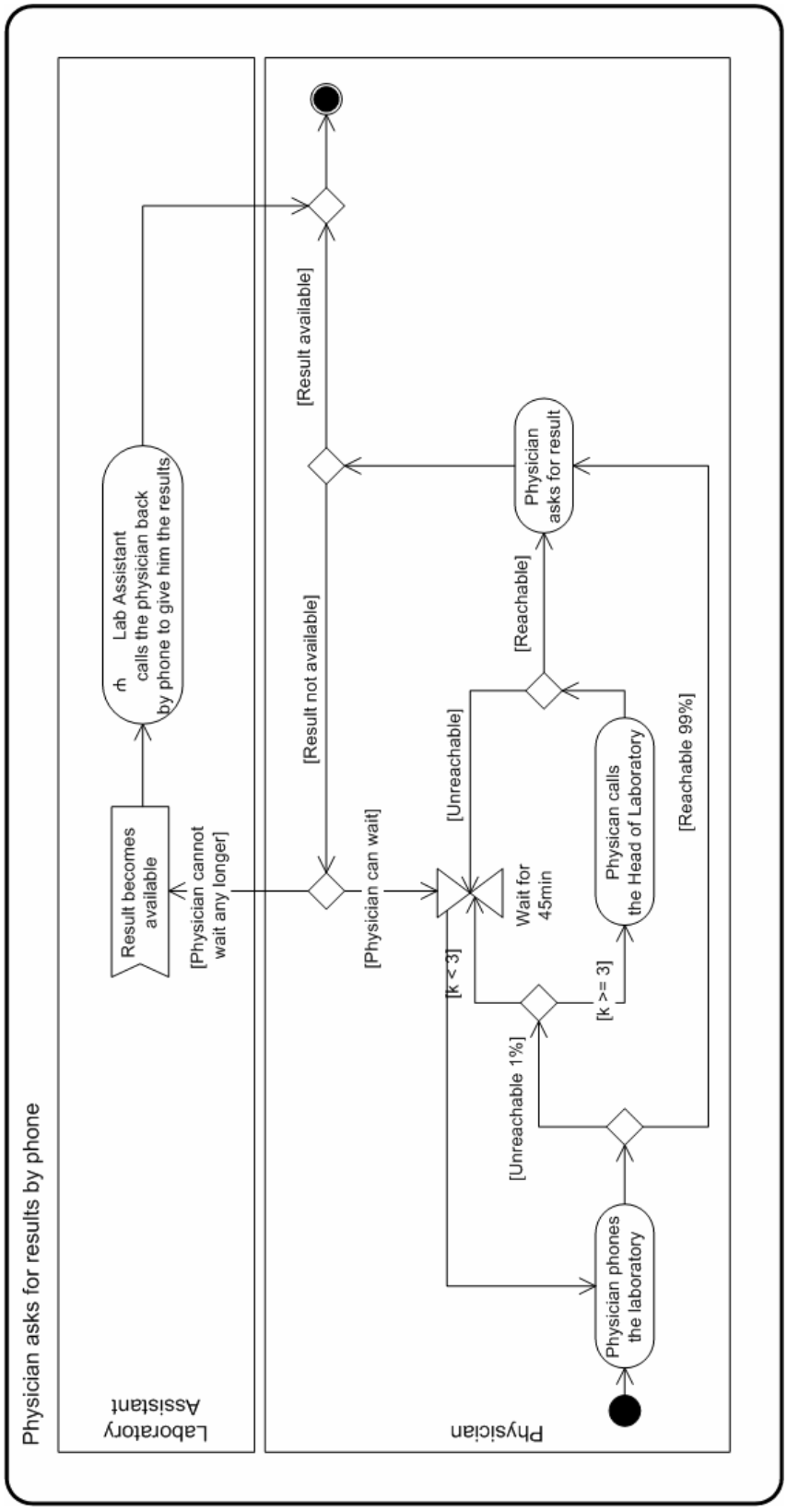

Figure 19. Physician asks for results by phone

A.3. Laboratory Assistant calls the physician back by phone to give him the result

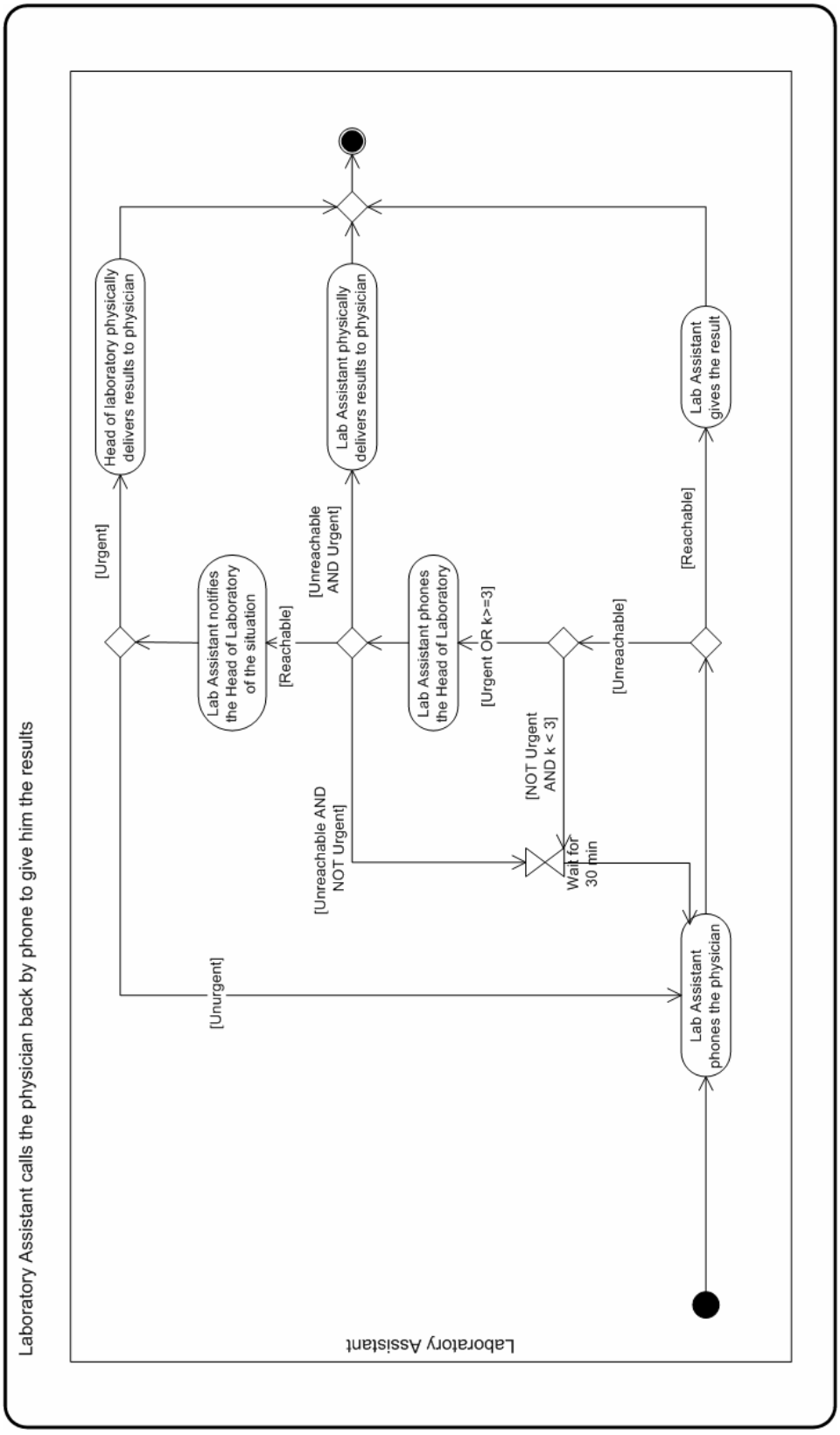

Figure 20. Laboratory Assistant calls the physician back by phone to give him the test results

Acknowledgements

The MediMAS case study could not have been performed without the help and support of Dr. Jean Luc Magnin, PhD, Head of the Laboratoryof the HCF, who took his precious time to explain to our research team the organization and the business processes of the laboratory of the HCF. We also thank Dr. Benoît Felley, PhD, for his valuable advice, and the laboratory personnel for the live demonstration of the business processes in the real world.

We also wish to thank Nguyen Dac Hoa, Dr. rer. pol., for his patient and careful proofreading of the draft manuscript. His advice and remarks helped us to enhance the readability of our presentation and to correct many mistakes. Needless to say, we are responsible for all errors that remain.

References

Acronymics, Inc., (2006). *Agent Builder – An Integrated Toolkit for Constructing Intelligent Software Agents.* **http://www.agentbuilder.com** (last accessed: December 06th, 2007).

Adolph, S., Bramble, P., (2002). *Patterns for Effective Use Cases*. Addison-Wesley Professional.

Agent Oriented Software Pty. Ltd., (2006). *JACK(TM) Intelligent Agents Agent Manual,* Release 5.2, **http://www.agent-software.com/shared/demosNdocs/Agent_Manual_WEB/index.html** (last accessed: November 17th 2007).

AOTLab, (2007). *The Agent and Object Technology Lab at University of Parma*, University of Parma, Italy. **http://aot.ce.unipr.it/** (last accessed: November 19th 2007).

Bellifemine, F. L., Caire, G., Greenwood, D., (2007). *Developing Multi-Agent Systems with Jade.* New York: J. Wiley.

Bellifemine, F., Poggi, A., Rimassa , G., (2001). *Developing multi-agent systems with a FIPA-compliant agent framework.* software—Practice and Experience Softw. Pract. Exper. 2001; 31:103–128.

Bisbal, J., Lawless, D., Wu, B., Grimson, J., (1999). *Legacy Information Systems: Issues and Directions*, Vol. 16, No. 5, pp. 103-111.

Bui, X. T., (2000). Building agent-based corporate information systems: *An application to telemedicine*. European Journal of Operational Research. v122. 242-257.

Datamed SA, (2007). *Informatique Médicale et Scientifique, WinDMLAB – la gestion de laboratoire éprouvée.* **http://www.datamed.ch** (last accessed: November 19th, 2007).

de Groot, D., Brazier, F. (2006*). Identity Management in Agent Systems.* in: Proceedings of the 1st International Workshop on Privacy and Security in Agent-based Collaborative Environments (PSACE) at the Fif th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-06), pages 23-34, Future University, Hakodate, Japan, 2006. **http://www.iids.org/**

Deloach, S. A., Kumar, M., (2005). *Multi-Agent Systems Engineering: An Overview and Case Study*. In: Henderson-Sellers, B., Giorgini, P. (Eds.), Agent-oriented methodologies, Idea Group Publishing. pp. 317-340.

Fasli, M., (2007). *Agent Technology for E-Commerce*. New York: J. Wiley.

Ferber, J., (1999). *Multi-Agent Systems*. Boston: Addison-Wesley.

FIPA, (2007). *IEEE Foundation for Intelligent Physical Agents*. **http://www.fipa.org** (last accessed: December 20th, 2007).

Franklin, S., Graesser A., (2001). *Is it an Agent, or just a Program? A Taxonomy for Autonomous Agents.* Proceeding of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

Gachet, A., P. Haettenschwiler (2005). *The Virtual Twin: A Socialization Agent for Peer-to-Peer Networks,* International Journal of Intelligent Information Technologies 1(2):56-67, 2005.

Giorgini, P., Kolp, M., Mylopoulos, J., Castrol, J., (2005). *Tropos: A Requirements-Driven Methodology for Agent-Oriented Software*. In: Henderson-Sellers, B., Giorgini, P. (Eds.), Agent-oriented methodologies, Idea Group Publishing. pp. 20-45.

Gozdan, S. W., (2007). *How big is your process automation gaps?* Mortage Banking. ECT News Network.

HCF, (2007). *Hôpital Cantonal de Fribourg*. **http://www.hopcantfr.ch/index_hop.html** (last accessed: December 20th, 2007).

Henderson-Sellers, B., Giorgini, P. (2005). *Agent-Oriented Methodologies*. Hershey: Idea Group Publishing.

IBM Research, 2002. *Aglets*. **http://www.trl.ibm.com/aglets/**

Iglesias, A. C., Garijo, M., (2005). *The Agent-Oriented Methodology MAS-SommonKADS*. In: Henderson-Sellers, B., Giorgini, P. (Eds.), Agent-oriented methodologies, Idea Group Publishing. pp. 47-78.

Lanzola G, Gatti L., Falasconi S, Stefanelli M., (1999). *A framework for building cooperative software agents in medical applications*. Artificial Intelligence in Medicine 16, pp. 223-249, 1999.

Luck, M., Ashri, R., and D'Inverno, M., (2004). *Agent-Based Software Development*. Artech House Publishers.

McCrickard, D. S., and Cheng, M. W., (1998). *How do we interact with an agent: A perspective from an undergraduate human factors class*, March 1998. Presented at the 1998 Human Computer Interaction Consortium, Available as GVU Technical Report GIT-GVU-98-29.

Nikraz, M., Caire, G., Bahri, PA., (2006). *A methodology for the analysis and design of multi-agent systems using JADE.* International Journal of Computer Systems Science and Engineering.

Opalis Software, Inc., (2007). *Closing the IT process automation gap,* **http://www.opalis.com/** (last accessed: Novembre 17th, 2007).

Ruppen A., (2007). *Système multi agents – MediMAS: Etude de cas dans le domaine du e-Health Care*, Bachelor thesis, Department of Computer Science, University Fribourg. **http://diuf.unifr.ch/softeng/student-projects/completed/ruppen/index.html**

Sommerville, I., (2006). *Software Engineering*. Harlow: Addison-Wesley.

SQLite, (2007). *SQLite RDBMS*. **http://www.sqlite.org/** (last accessed: December 20th, 2007)

Stanford Center for Biomedical Informatics Research, (2007). *Protégé*. **http://protege.stanford.edu/aboutus/aboutus.html** (last accessed: December 02nd, 2007).

Telecom Italia Lab, (2007). JADE – *Java Agent DEvelopment Framework*, Version 3.5, **http://jade.tilab.com/**

Telecom Italia Group, (2007). *Telecom Italia Lab*. **http://www.telecomitalia.com/** (last accessed: November 17th 2007).

TopQuadrant, Inc., (2007). TopBraidComposer. **http://www.topbraidcomposer.com/index.html** (last accessed: December 17th, 2007)

Unland, R., Klusch, M., Calisti, M. (2005). *Software Agent-Based Applications, Platforms, and Development Kits*. Basel: Birkhäuser Verlag.

van Aart, C.J., (2007). *Ontology Bean Generator*. **`http://protege.cim3.net/cgi-bin/wiki.pl?OntologyBeanGenerator.`** (last accessed: December 17th, 2007)

Weiss, G., (1999). *Multiagent Systems*. Cambridge: MIT.

Wooldridge, M., (2002). *An introduction to MultiAgent Systems*. London: J. Wiley.

Walter J. D., et als, (1998). *Handbook of Health Care Management.* Blackwell.

Wooldridge, M., Jennings, N. R., Kinny D., (2000). "The Gaia methodology for agent-oriented analysis and design," *Autonomous Agents and Multi-Agent Systems*, vol. 3(3), pp. 285-312, 2000.

Wooldridge, M., Bussmann, S., and Jennings, N. R., (2004). *Multiagent Systems for Manufacturing Control*. Springer-Verlag.

Zambonelli, F., Jennings, R.N., Wooldrige, M., (2005). Multi-agent systems as computational organizations: *The Gaia methodology*. In: Henderson-Sellers, B., Giorgini, P. (Eds.), Agent-oriented methodologies, Idea Group Publishing. pp. 136-171.