Using Agent Technology for Exception Monitoring

The Case of Fraud Detection in Conference Management Systems

MASTER THESIS

Sedef Z. Sicakkan

Department of Information Science and Media Studies

University of Bergen

Fall 2010

Acknowledgements

I owe special thanks to my adviser, Weiqin Chen, for her through readings of the manuscript, her detailed critical remarks and crucial suggestions, and her kind support. Her AI and programming courses have been an inspiration source in the formulation of the main research questions of this project.

I would like to express my gratitude to my former colleagues at Unifob Global, IMER-UiB and the EUROSPHERE-program where I had the opportunity to observe the specific business problems associated with the organization of academic conferences in an intellectually stimulating international academic milieu.

My interview subjects also deserve thanks for participation and their useful feedback during the testing of MONA.

I would also like to thank my dearest ones: Hakan, Cengiz , Selina, Bilgen, Ismail and Ayse for their support in the process of completing this thesis. Their presence and the loving memory of Kadriye and Tului have always been an inspiration.

The process of writing this thesis had to be extended due to the birth of my wonderful Selina, and it is completed while I am working at a full position. I believe that every input helped me to improve the project. However, the responsibility for any erroneous result remains mine.

Abstract

Exception monitoring involves solutions to overcome the fatality of exceptions. The general idea behind this concept is that exceptions should be monitored, and as soon as they occur, they should be eliminated in one way or another so that the systems keep on working without being impacted by the potential harm of these unusual situations present. Misuse or fraud is one of such exceptional situations. In this thesis, a monitoring agent application, MONA, is designed, developed and evaluated in order to support business processes for capturing fraudulent activities with an empirical focus on the case of conference management systems. With the help of this prototype, also the time and quality impact of agent technology, and if agent technology creates an increase in time effectiveness and enhances the quality of work, are investigated. Another focal point of the project is to find out if the results of collaborating with the monitoring agent are reliable, and whether they are convincing enough for further collaboration. In the evaluation process usability test is conducted through observations and interviews with the potential users. The results indicate that there are gains in time, time-effectiveness increases, and the quality of fraud detection work is enhanced. Another important result is that the test participants are likely to trust MONA.

Table of Contents

List of Tables

List of Figures

Prototype

MONA: http://www.ocs.uib.no/Mona/index.php

Source code: http://sourceforge.net/projects/monitoringagent/

Open Conference Systems

MONA uses OCS version 2.1.1-2: http://www.ocs.uib.no/thesis/ocs/

More information about OCS: http://pkp.sfu.ca/?q=ocs

1 Introduction

Earlier research has indicated that using information systems with well-represented business needs is a key factor in successful business. Such systems improve or dramatically alter the work organization, production and almost every process that is connected to the business itself. Briefly, investments on technological capital yield a return in the form of improved business processes and thus help boost revenues.

This ideal picture, however, has some (vital) exceptions in the real world. The most prominent ones are mostly connected to the human factor: *users* of the information systems. Although business needs are well-represented in information systems and systems work effectively and efficiently, it is seldom possible to perfectly shield a system from misuse or fraud. Besides, most such systems need monitoring tools. Thus, it is very important that misuse and fraud are captured in an early phase of development processes and represented in the information systems (Wilhelm, 2004).

The claim of this thesis is that *fraud is an exception* in the sense I have sketched out above. First, I want to show how it is possible for even well-established information systems to allow fraudulent actions through those systems' own set of business rules and workflow processes. Second, I want to investigate whether an agent application will be useful in supporting business processes for capturing such fraud, with an empirical focus on the case of conference management systems.

1.1 The Possibility of Fraud in Conference Management Systems

Most conferences use *conference management systems* (CMSs), which are used for two main purposes: to disseminate information about conferences and calls for papers and to collect information about registrants, abstracts, papers etc. In other words, any information related to a conference can be disseminated or collected with the help of such web-based systems. CMSs provide easy-to-use tools for conference attendees and conference managers so that an attendee can register him- or herself to a conference or a conference manager can follow the embedded workflow[1] processes, which consist of e.g. tasks and activities that should be completed by conference attendees. The conference attendees benefit by being able to skip

[1] Huhns and Singh (1998) define workflow as "a composite activity consisting of tasks involving a number of humans, databases, and specialized applications".

most of the processes that need extra attention and communication with different organizations since all these communication and cooperation demands are somehow supplied by the system (e.g. sending personal information, fee payment, accommodation etc.). As for the conference managers, their business processes become more seamless and less costly. Rather than having to take on certain sub-processes themselves, such as collecting registration details manually, they can monitor the status of processes with the available tools and retrieve the necessary information they need from the system.

However, the seamlessness of processes in a conference management environment has some disadvantages as well. Business processes in CMSs can be misused as a cover for shady activities if there are no safeguards against this. It is assumed that the rules embedded in a system are legal and, when the process is completed, the results will be legal as well. In some cases, even legal processes can open up the possibility for illegal acts. As a result, it is necessary to have tools that enable systems to monitor and detect fraud attempts effectively.

The function of such tools should be to extract relevant information from the database, and present the results to organizers in an accurate, structured and intelligible way, so as to enable them to detect fraud immediately and take decisions effectively in the event of fraud.

For conference managers, it is important to know what kind of information the *conference database* contains and how comprehensive and consistent the information is. CMSs come with some reporting options that are usually good enough to provide reports of aggregate information stored in the database. Although the information is already stored in the database, such systems lack tools that can be useful to detect fraud immediately and effectively. It is not always the case that conference managers have hands-on experience with their respective databases. Furthermore, it is time- and resource-demanding to check the records manually. As the scale of the conference expands and the number of attendees grows, so does the difficulty of detecting fraudulent activities. This makes manual checks even harder and less realistic.

1.2 Scenario

For the purposes of this introduction, it will suffice to illustrate this difficulty with a single example. In international academic conferences, it is expected that there will be registrants from all over the world. In order to travel across international borders, it is necessary to have the proper documentation, i.e. a valid passport and visa. According to the regulations in

Norway, for instance, foreigners who want to get a visa should apply to the Norwegian embassies in their own countries. Usually, they are asked to present an invitation letter, proving that they are going to do what they claim to do in Norway. At this point, the possibility of fraud enters the picture: some conference managers have experienced that there are people who claim to be attendees in order to obtain an official invitation letter, which in turn will be used to obtain a visa to enter the country. Based on the interviews I have conducted with some conference managers, this is a known case, and managers of international conferences have to work especially hard to eliminate such fraud attempts by having extra checks. Therefore, data collection is very crucial in registration processes since these processes supply the raw material for extra checks.

I will here focus on payment information, since among the data that gets stored in CMSs; payment information provides useful hints for discovering possible fraud attempts. The normal practice is that everybody pays for him- or herself, and not on behalf of another person. In the event that (as a result of insufficient resources) one person pays the conference fee on behalf of several others, these attendees are expected to get in touch with the conference organization committee in one way or another, and the CMS should also notify the conference manager. If these attendees do not get in touch with the conference manager, then two possible scenarios (among others) can occur. In the first, which is a hypothetical situation, a person (perhaps a human trafficker) pays the full conference fee for several people. The second scenario, which has been observed, involves a hacker creating several "dummy" registration data at a time close to the registration deadline of an international conference. In both of these cases, the conference management system lacks monitoring tools; rooting out fraudulent attempts manually becomes a huge task to accomplish since (at the very least) account details should be checked against the conference registrations, and it might take some days to clean the registration data and unearth the real attendees of the conference.

1.3 Proposed Solution

The concept of *fraud* includes the improper use of systems to make an illegal asset. The solutions and research encompass a wide range of business and fraud types such as insurance payments for car repairs (Aart & Tamma, 2008), money laundering (Liu & Zhang, 2007), electronic fraud (Prodromidis & Stolfo, 1999), and credit card fraud (Chan, Fan, Prodromidis, & Stalfo, 1999).

Based on the scenario sketched above, a system which monitors possible fraudulent activities is necessary to support this kind of conference management system, which lacks tools for monitoring. It is important that this system fulfils the relevant tasks without human intervention so that conference managers are able to focus on the results generated by the system, rather than using time doing the job manually. Therefore, "the system to be developed requires its components to show a high degree of autonomy" (Caire, Coulier, Garijo, Gomez, Pavon, Leal, et al., 2002).

Another aspect is related to the functions this system will perform. Finding possible fraudulent activities is actually composed of several tasks. The system to be developed should do the necessary checks on behalf of the conference manager by taking on his or her workflow items and flagging the possible fraud attempts. The conference manager should make decisions on whether actions needed to be taken. Thus, this system should work as an assistant to the conference manager to help label possible fraud attempts in a registration process. Since there are different tasks and there is a hierarchy in the division of labour, "goal-oriented behaviour will be needed" (Caire, et al., 2002).

As a result, a monitoring agent can be useful in obtaining information about fraud. The agent will take an active part in highlighting the attendees, who might be using the conference in order to enter the host country illegally rather than attending the conference. In short, the monitoring agent will accept task(s) and produce easy-to-understand outputs that can support the manager's work in rooting out the possible fraud attempts in the registration process.

1.4 Research Questions

This project seeks to answer the following research questions:

- Can a monitoring agent tool be useful in supporting conference managers' efforts to detect fraud in CMSs?
- What is the impact of agent technology integration on time and quality in medium-scale conference management systems? Does agent technology increase time effectiveness and enhance quality?
- Are the results of collaborating with the monitoring agent reliable? Are they convincing enough for further collaboration?

1.5 Research Design

This project's research design is based on the design-science paradigm, which "is fundamentally a problem-solving paradigm" (Hevner, March, Park, & Ram, 2004). In this paradigm, "knowledge and understanding of a problem domain and its solution are achieved in the building and application of the designed artefact" (Hevner, et al., 2004).

This thesis centres on the design and development of a monitoring agent that will support workflow processes in detecting fraud in a conference management system. In order to achieve this, four different research activities are conducted:

1. Identification of the core manifestations of fraud based on the observed and hypothetical cases of fraud attempts (interviews with experienced conference managers)
2. Design and implementation of a prototype
3. Evaluation of the working-prototype through usability testing
4. Identification of the benefits of using agent technology for monitoring fraudulent activities in CMSs (based on usability testing)

1.6 Organization of the Thesis

This thesis is organized as follows:

Chapter 1 gives a brief introduction to the problem and introduces the scope of the project by defining problem scenarios, proposed solutions, associated research questions, and a research design.

Chapter 2 focuses on the construction of the project's theoretical and conceptual framework. There are four major issues taken up in this chapter. The first is "Agent definition", where the concept of "agent" is discussed. Particular focus is placed on monitoring agents and related work. Then, the core information about software processes and agent-oriented development methodology is given, and there are short discussions on the chosen techniques. In the last section of this chapter, the evaluation of monitoring agents is explained briefly with a discussion of evaluation techniques.

Chapter 3 explains and discusses topics related to design and development. This chapter departs from the agent-oriented approach and introduces the notion of agent domain, before focusing on the modelling of the monitoring agent for *Open Conference Systems* (OCS). The analysis of the problem domain is followed by the actual design of the MONA (**MON**itoring **A**gent for fraud detection in OCS). The topics that are related to the development of the prototype include the integration of MONA with OCS environment, architecture, and software development that is organized in iterations.

Chapter 4 is dedicated to the evaluation of the prototype and research findings. It details how the usability test is conducted and the results, along with the answers to the research questions.

The final chapter of this thesis is devoted to conclusions and future work.

The thesis has two appendices. The first appendix includes all the documents (informational texts, survey forms, test data and interview questions) used in the usability test. The second appendix presents the user manual for MONA.

2 Constructing the Theoretical and Conceptual Framework

Agent technology and exceptions in the workflow in information systems are the two central concepts of this research project. What connects them is the idea of integrating an exception monitoring agent into a conference management domain.

In this chapter, I will give brief presentations of different conceptual approaches to agents and agent development. Here the aim is to introduce the design approach adopted in this project in a theoretically well-situated context by positioning it within existing approaches.

In the following, the focus will be on agent technologies, covering central issues from the concept of an agent to the classification of agents. In particular, I will describe monitoring agents, and I will give some examples of currently existing monitoring agents and exception monitoring agents. Moreover, a framework for software development approaches and agent-oriented development will be given briefly. Finally, I will discuss evaluation methods.

2.1 Approaches to Agent Definition in Previous Literature

2.1.1 What is an agent?

Nwana (1996, cited in Bradshaw, 1997) makes a distinction between two strands of agent research based on the history of the field, "the first beginning about 1977, and the second around 1990." The former can be considered to have roots in distributed artificial intelligence (DAI), and "has concentrated mainly on deliberative-type agents with symbolic internal models." This line of research mainly works on "macro issues such as the interaction and communication between agents, the decomposition and distribution of tasks, coordination and cooperation, conflict resolution via negotiation, etc." The latter focuses mainly on "doing" and "remote action". In this project, I will follow the latter course which was established in the 90s[2] since my project's research questions are based upon what the agent is "doing" and how it is "done".

Although "one of the key concepts of agent-based computing" (Jennings, 2001) is an *agent*, there is not "a single universally accepted definition" (Wooldridge & Jennings, 1995)

[2] Wooldridge (2009:393–404) mentions this period in his work on the history of intelligent agent research.

for this term. However, various definitions intersect at some core features which have shaped the general understanding of the concept. As mentioned in Padgham and Winikoff (2004: 1), the following definitions are adapted from Wooldridge and Jennings (1995):

> An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

Agent-systems have the following properties:[3]

Autonomous: Agents have control both over their internal state and behaviour.

Situated in a particular environment: Agents have partial control and observability in this partial environment.[4]

Having a specific role: Agents are designed to fulfil a specific role; that is, they have particular objectives to achieve.

Flexible problem-solving ability: Agents have the capacity to solve problems with well-defined boundaries and interfaces. What lies behind the need for being flexible is that "agents must be robust" (Padgham & Winikoff, 2004: 3) enough to recover from various failures.

Reactivity: Agents are able to respond to changes in their particular environment.

Proactivity: Agents can take initiatives according to their predefined goals.

Social ability: Agents interact with other agents, including humans, via some kind of *agent-communication language* (Genesereth & Ketchpel, 1994, cited in Wooldridge & Jennings, 1995).

Wooldridge and Jennings (1995) consider the above-mentioned features to be a "weak notion of agency."[5] They claim that a "stronger notion of agency" comes with *having mentalistic*

[3] There is broad agreement in the literature about what agent properties are: Binder, Mori, Portabella, Tamma, and Wooldridge (2005:1-4), Grau, Cares, Franch, and Navarrete (2006), Jennings (1999, 2001), Jennings, Faratin, Norman, O'Brien, and Odgers (2000), Kishore, Zhang, and Ramesh (2006), Luck, McBurney, and Preist (2003:10), Tveit (2001), Wooldridge (1997, 2002), Wooldridge and Jennings (1995), and Wooldridge and Ciancarini (2001).

[4] In Giorgini and Henderson-Sellers (2005), this is called "situatedness". There is literature that deals with situation awareness and proactivity, such as So and Sonenberg (2004) and Urlings, Tweedale, Sioutis, and Ichalkaranje (2003). Padgham and Winikoff (2004:1) define a typical agent environment as being dynamic in that "they change rapidly", unpredictable in that "it is not possible to predict the future states of the environment" and unreliable in that "the actions that an agent can perform may fail for reasons that are beyond an agent's knowledge and influence."

notions such as “knowledge, belief, intention, and obligation” (Bates, Loyall, & Reilly, 1992; Bates, 1994, cited in Wooldridge & Jennings, 1995) and add the further criterion of *having human-like emotions*. Shoham (1997) defines an agent precisely as “an entity whose state is viewed as consisting of mental components such as beliefs, capabilities, choices, and commitments.”. Among other features of agency, the following features are listed in Wooldridge and Jennings (1995):

> *Mobility*: Agents can move around an electronic network (White, 1994, cited in Wooldridge & Jennings, 1995).
>
> *Veracity*: Agents will not knowingly communicate false information (Galliers, 1988:159-164, cited in Wooldridge & Jennings, 1995).
>
> *Benevolence*: Agents do not have conflicting goals, and every agent will therefore always try to do what is asked of it (Rosenschein & Genesereth, 1985, cited in Wooldridge & Jennings, 1995).
>
> *Rationality*: Agents will act in order to achieve their goals. As far as their beliefs permit, they will not act in such a way as to prevent their goals (Galliers, 1988:49-54, cited in Wooldridge & Jennings, 1995).

In recent works (e.g. Bernon et al., 2005a), the essential properties of agents are listed as: autonomy, proactivity, the ability to communicate with other agents and the ability to perceive its environment.[6] In such definitions, continuity over time is also included either explicitly or implicitly; that is, the agent will be functioning continuously as long as there is a reason for it to do so “without requiring constant human guidance or intervention” (Bradshaw, 1997).

The usefulness of the agents is testified by the success of implementing the above-mentioned features. In Padgham and Winikoff (2004:4–6), these features are defined as follows: autonomy, agents’ having control over their goals and states, reactivity, proactivity, flexibility and the ability to deal with changing environments effectively. In the view of Padgam and Winikof (2004:4-6), this is why agents fit better in wrapping legacy software. They claim that

---

[5] Menzies, Pearce, Heinze, and Goss (2002) use “weak agents” and claim that “weak agents maps neatly into object technology.”

[6] As mentioned in this work, these “properties were defined during the second meeting of the AgentLink3 AOSE TFG (Ljubljana, February 2005).”

agents are proactive and reactive so that they have the ability to solve the problems just like humans do.

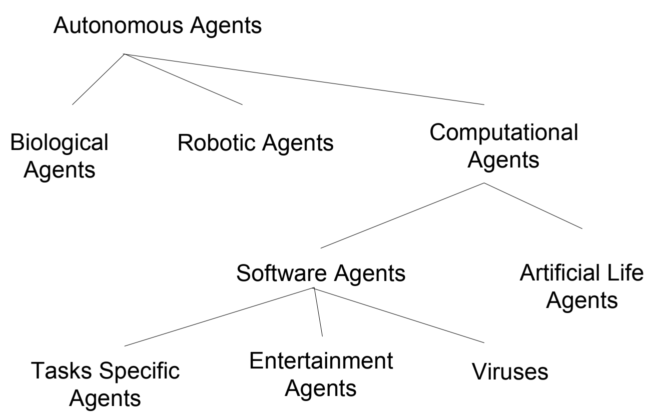


Figure 1: **Autonomous Agent Taxonomy**
***Source*: Franklin and Gaesser (1997)**

There are many different agent taxonomies in the existing literature. Here, I will focus on Franklin and Graesser's (1997) taxonomy. They classify agents according to their properties such as reactivity, autonomy, goal-orientation, temporal continuity, communication, learning, mobility, flexibility, and having a character. Thereby, they suggest the taxonomy of autonomous agents as shown in Figure 1.

2.1.3 Agents vs. Objects

As stated in Shoham (1997), "a computational framework called Agent-Oriented Programming (AOP)", which "can be seen as a specialization of the Object-Oriented Programming (OOP) paradigm", has been proposed to highlight the "cognitive and societal view of computation".

What makes agents different from objects? Or, as some researchers have asked, "may agents be considered as an extension of objects and then classical object-oriented software engineering can be used as well to build agent-based applications?" (Bernon, Cossentino, & Pavón, 2005b).

Although agents and objects are alike in some aspects ("there are clearly close links between agents and objects" [Wooldridge & Ciancarini, 2001]), it is these same aspects that make them different from each other and put agents a step ahead[7].

There are two main notions that draw a clear line between agents and objects (Bernon et al., 2005b; Wooldridge & Ciancarini, 2001): autonomy and interaction. *Autonomy* is tightly connected to the idea of "having a state". Here, it is assumed that "having a state" implies

[7] Bernon et al. (2005b) support this idea, and claim that "the use of object-oriented software engineering techniques can be applied for the development of MAS [Multi Agent Systems]"; on the other hand, some extensions are necessary to support the essential properties of agents (Wooldridge & Ciancarini, 2001).

having control over one's own state. An object's state is reflected by its attributes or methods, where it has limited control over its own state or "behaviour" (Wooldridge, 2009:29). Regarding an agent, its state includes more than having attributes or methods, such as its beliefs, goals and tasks are embedded in its state—which also adds *flexibility* into the notion of autonomy. That is why it has been frequently mentioned that agents have mental states. "This distinction between objects and agents has been nicely summarized in the following slogan: 'Objects do it for free; agents do it because they want to'" (Wooldridge & Ciancarini, 2001; Wooldridge, 2009:29). Another important point is that agents "have their own thread of control" (Wooldridge & Ciancarini, 2001; Wooldridge, 2009:29-30); that is, objects are passive while agents are active (Bigus & Bigus, 2001:8)[8] and agents are capable of controlling their own states. This separation is based on how they behave: Objects should be called by an external actor, but agents can be "engaged in an infinite loop of observing their environment, updating their internal state, and selecting and executing an action to perform" (Wooldridge & Ciancarini, 2001). The "notion of autonomy is stronger in agents" (Bernon et al., 2005b; Wooldridge & Ciancarini, 2001): because agents have ability to determine their behaviour according to their states, including their mental states. Agents are active entities (reactive) and can take initiatives (proactive) "depending on its goals, its internal state and its knowledge from the environment" (Bernon et al., 2005b).

Another major difference between agents and objects is that of *interaction*. The interaction between objects is limited to messages, which invoke methods. However, in an agent world, interaction includes co-ordination, organization, negotiation and cooperation, all of which amounts to the agent's being interactive within its environment. Agents are considered to be "social entities [which] communicate and interact with other entities that share a common environment" (Bernon et al., 2005b; Wooldridge & Ciancarini, 2001), and "agents have conversations" (Bigus & Bigus, 2001:8). In other words, *agents are alive in their environments*. However, "the standard object model has nothing to say about such types of behaviour [reactive, proactive, and social]" (Wooldridge, 2009:30).

[8] "[A]n event is anything that happens to change the environment or anything of which the agent should be aware.…[W]hen an event does occur, the agent has to recognize and evaluate what the event means and then respond to it."

2.2 Monitoring Agents

In the classification of Franklin and Graesser (1997), monitoring agents are "task specific agents". Lee (2000) takes this classification a step further by classifying them as *reactive systems* based on their specific behaviour in dynamic environments. This classification is based on the properties of the environment, which is often *dynamic*, *unpredictable* and *unreliable*, and the behaviour that these agent systems represent.

Since agents are situated in a particular dynamic environment, it is important that they react effectively to this rapidly changing environment. By being in an unpredictable environment, it is assumed that agents do not have control over their environments; in other words, they cannot predict the future states of their environment. Being unreliable indicates that agents can fail for reasons beyond their own control. However, it is also expected that they should be flexible enough to exhibit different solutions to the same problem, and that perhaps the recovery should be obtained automatically so that the changes in the environment do not keep agents from reaching their goals. Besides, agents have the ability to communicate with human (agents) via an agent language or well-designed GUI (Graphical User Interface), where they might be perceived as having humanlike responses. In some cases, agent systems are considered to be human-substitutes. Unlike humans, however, they have the advantage of not becoming bored of doing the same job as many times as necessary without losing effectiveness or increasing error rates. Thus, the features of an agent are well suited for automated monitoring or controlling systems which demand "dynamic movement of time-critical tasks among software and human agents" (Lee, 2000).

2.2.1 Existing Monitoring Agents

Various kinds of monitoring agents are presented in the existing literature. They are mainly used in monitoring processes in different areas such as power plants (Heo & Lee, 2005), complex chemical processes (Bunch, Breedy, Bradshaw, Carvalho, Suri, Uszok, et al., 2004), online diagnostics, i.e. condition/patient monitoring diagnostics (McArthur, Booth, McDonald, & McFadyen, 2005; Mabry, Schneringer, Etters, & Edwards, 2003; Uckun, 1994), voice quality management (Imai, Yamada, Ueno, Nakamichi, & Chugo, 2006), academic management (Jami & Shaikh, 2007), transport (Jedermann & Lang, 2006), detecting money laundering (Wang, Xu, Wang, Ye, & Gao, 2007), and workflow monitoring (Wang, Wang, & Xu, 2005; Ehrler, Fleurke, Purvis, & Savarimuthu, 2006). Monitoring agents "help in these types of tasks because they do not get bored when nothing is happening, and during crises they can help to manage information overlaid" (Hayes, 1999).

An interesting notion is related to their ability to take part not only in particular processes but also in monitoring intelligent multi-agent systems. In Kaminka, Pynadath, and Tambe (2001), these agents are used to monitor the existing intelligent systems, such as "on-line monitoring of deployed distributed teams of cooperation agents, e.g., for visualization, or performance tracking".

The earlier research contains different names for monitoring agents: "user agents", "assistant agents" or "interface agents". These agents have slightly different roles, but they all do some kind of monitoring activity. For example, Lashkari, Metral, and Maes (1998) and Maes (1994, 1997) describe a single-user *interface* agent (Wooldridge, 1997) which filters emails on behalf of the user. Basically, the user agent's mission is to interact with the application, communicate with the human agent, monitor the user's actions and learn from them so that the agent can imitate these actions. Luck, McBurney, and Preist (2003:26) describe an assistant agent as one that is "engaged in gathering information or executing transactions on behalf of their human principals on the Internet".

2.2.2 Exception Monitoring Agents

*Exception*s can be defined as *unusual situations* which can be "special cases or error conditions".[9] Special cases are those that somehow do not fit the logic of the processes, although such exceptional cases might be a part of the processes. On the other hand, error conditions might occur due to the interruption of the flow of the programs. It is important that the system have the capacity to deal with them, since otherwise, the system may break down when exceptions occur.

Exception monitoring involves solutions to overcome the fatality of exceptions. The general idea behind this concept is that exceptions should be monitored, and as soon as they occur, they should be eliminated in one way or another so that the systems keep on working without being impacted by the potential harm these unusual situations present.

In the previous literature, there are many examples of agent systems that can successfully deal with exceptions.

[9] www.stanford.edu/class/cs242/readings/vocabulary.html (Last accessed 9.february.2010)

When business processes are the sources of exceptions, it is necessary "to predict and prevent business process exceptions as early as possible before they occur, and detect and resolve exceptions as soon as possible after they occur" (Kim, Choi, & Park, 2010). This approach is known as proactive exception handling. Another option, reactive exception handling, involves dealing with the "unexpected output of the tasks" (Wang & Wang, 2004).

It is also possible to classify exception monitoring together with "anomaly detection". For example, McArthur, Booth, McDonald, and McFadyen (2005) demonstrate "an agent-based architecture [which] can support anomaly detection for condition monitoring of [an] electrical plant".

Another case worth mentioning is that of enhancing the monitoring system's flexibility in order to deal with exceptions. For example, Wang and Wang (2002) propose a flexible workflow monitoring system which allows for exception management.

With respect to error cases, one solution might be monitoring the target system, which might be impaired because of exceptional situations. Hu, Zhang, Zhang, Zhao, Chen, and Fang (2008) give an example of "an agents group-based approach" for exception handling in which an "agents group is dispatched to find out the status of running processes in the system to keep track and troubleshoot them when necessary". Another example is the proposal in Yang, Chan and Xu (2005) of "a mobile agent-based approach to handling exceptions in distributed workflow management systems".

2.3 Software Process and Agent-Oriented Development

In building an exception monitoring agent, it is important to follow an agent-oriented development methodology "which explicitly model[s] agent-oriented aspects" (Bussmann, Jennings, & Wooldridge, 2004:80). In this section, I will review the current software development approaches, before focusing on agent-oriented development methodology. As mentioned above, the aim of this exercise is to explicate and theoretically situate the approach that is adopted in this project.

2.3.1 Software Process Approaches

There are two widely used software process approaches in the literature: *the waterfall model* and *iterative and evolutionary development*.

2.3.1.1 The Waterfall Model

This approach departs from the idea that the software development process is composed of distinct stages. It “follows a logical progression of defining requirements, designing the system, building the system, testing the system, and placing the system into operation” (Perry, 2006:587). Each of these stages has its own type of activities, and it is necessary to produce documentation at the end of each stage. As a result of this necessity, the number of iterations is limited in this model, since every iteration comes with a higher level of costs that are related in particular to “the cost of producing and approving documents” (Sommerville, 2004:66-68) as well as the fact that every iteration involves “significant rework” (Sommerville, 2004:66-68). On the other hand, the lifeline in this model is linear, “but there are often backflows” (Fowler, 2004:20). These backflows are not part of the routine; they are considered “exceptions and should be minimized as much as possible” (Fowler, 2004:20). Moreover, waterfall methodology “assumes that at the end of the requirements phase, requirements for development are known” (Perry, 2006:587). Therefore, “the commitments must be made at an early stage” (Sommerville, 2004:66-68). This leads to the fact that its response to changing customer needs is poor.

2.3.1.2 Iterative and Evolutionary Development Approach

The iterative and evolutionary development approach “breaks down a project by subsets of functionality” (Fowler, 2004:20). It “is based on the idea of developing an initial implementation, exposing this to user comment and refining it through many versions until an adequate system has been developed” (Sommerville, 2004:68-69). Its chief advantage is “the specification [that] can be developed incrementally” (Sommerville, 2004:68-69); this means that, it responds to changes in customer requirements in a better and faster way. This approach supports several releases, and each of these releases can be divided into several iterations. It is expected that each iteration produces “production-ready integrated software” (Fowler, 2004:20). In this way, the value from the system is received as early as the first iteration, and the feedbacks that are obtained are of a higher quality.

The nature of this development approach works well with projects’ time constraints. Backflows between stages are allowed and the software production is organized in different releases and iterations. Changes can be implemented as soon as possible and it is not necessary to make commitments early in the project timeline. This allows for working according to the priorities of the projects, and it also helps to identify the “real requirements priorities” (Fowler, 2004:20).

Since the system is produced incrementally in a feature-by-feature fashion, the development model requires the participation of the customer until the product is completed. It is assumed that “the user does not have a rigid definition of requirements” (Perry, 2006:587); the model instead takes its set of requirements from the final system in order to “provide feedback on its suitability and acceptability” (Sommerville, 2004:69). As a result, this type of software development approach is categorized as exploratory and throwaway or fast prototyping.

Even though the iterative and evolutionary development approach has many advantages over the waterfall model, Sommerville (2004:69) lists some of its disadvantages.

First of all, the software development in this approach happens so quickly that it might have a negative impact on the measurement of progress. Quick development processes can result in poorly structured systems, so that “incorporating further software changes becomes increasingly difficult and costly” (Sommerville, 2004:69).

Moreover, Sommerville (2004:69) claims that this approach is better suited to small and medium-sized systems; for large systems, he recommends a mixed process that “incorporates the best features of” both approaches.

Finally, Liskov and Guttag (2001:258) state that fast prototyping is suitable for simple systems, and it can be disadvantageous for complex systems. This is because such prototypes include a set of requirements that might not adequately reflect a system’s complexity. In addition, fast prototyping can lead to a system which is not robust. Furthermore, the production costs might run so high that the prototype “can be too valuable to throw away” (Liskov & Guttag, 2001:258).

2.3.1.3 The Choice of Software Process Approach

The iterative and evolutionary development approach will best serve the specific objectives of this project. This is a research project of an exploratory nature. The project’s research design is based on design-science paradigm, in which the solution to a problem includes the design and development of a working prototype which constitutes a *proof of concept*. The project’s exploratory nature is further bolstered in that by allowing changes in the requirements, the development process becomes self-enriching so that the findings can also be adapted to the prototype if necessary; that is, the feedback provided from iterations can be used in refactoring the prototype. This approach also allows for gradual development, so that the working prototype will verify the functional needs and business objectives will be implemented based on priority. Moreover, it better fosters the elimination of risks, since flexibility is embedded in the process. In building this prototype, it is expected that any problems (associated with the chosen tools, programming languages etc) can be eliminated relatively faster and more effectively.

2.3.2 Agent-Oriented Design and the Choice of Development Methodology

Agent-oriented development methodologies are "intended to assist first in gaining an understanding of a particular system, and, secondly, in designing it" (Wooldridge, 2009:184). There are well-known agent-oriented development methodologies such as AIII (the Australian AI Institute), Gaia, Tropos and Prometheus (Wooldridge, 2009:184-188).

Given the development tasks at hand, the *role-based* methodology is best suited to the scope of this thesis. Kendall (1998, 2001, cited in Bussmann et al., 2004:88) defines *role* as "an abstraction of agent behaviour based on responsibilities, possible collaborators, required expertise and cooperation mechanisms used". According to this methodology, as represented in GAIA, the analysis moves from abstract conceptualizations to concrete concepts (Wooldridge, 2009:186). Kendall states that the agent system is understood "as a process of organizational design". Therefore, the hierarchy can be represented with the concept of role, which is defined by "responsibilities, permissions, activities, and protocols" in GAIA (Wooldridge, 2009:187). These are precisely the features this project requires, for the following reasons:

First, the exception monitoring agent should provide different *roles* that support the monitoring *capabilities* based on the nature of its environment.

Second, in this particular conference management environment, *exceptions* emerge from the gap between the existing business processes and the implemented versions of these processes. To deal with it, the agent should have the capability to behave according to goals and communicate with human agents.

Third, it is important to emphasise the necessity of establishing a hierarchy between the monitoring agent and the human agent: The monitoring agent should gather information from the software environment to support the human agent's decision processes. This implies that the human agent is the one responsible for taking decisions regarding fraudulent activities.

Finally, the monitoring agent should have certain properties such as autonomy, reactivity and proactivity. By being autonomous, the monitoring agent should reach her goal and complete her tasks without human intervention, and she should be aware of her state. By being reactive, the agent has the ability to respond to the changes in her environment. Since there should be a hierarchy between agents (human and exception monitoring), the monitoring agent can also be assigned some tasks. This means that the agent should respond positively to the human agent's task allocation. She should act proactively; that is, she should take some initiatives that also support the monitoring activities in a rapidly changing environment.

2.4 Evaluation of the Monitoring Agent

According to the design-science paradigm, "IT artefacts can be evaluated in terms of functionality, completeness, consistency, accuracy, performance, reliability, usability, fit with the organization, and other relevant quality attributes" (Hevner, March, Park, & Ram, 2004).

In this thesis, the implemented prototype's usability will be evaluated. The main evaluation objective is to show whether a monitoring agent tool can be useful in supporting the workflow processes for capturing fraud for conference management systems.

2.4.1 Usability Testing

There are two aspects related to usability in this project: how to support usability in the design process and how to evaluate usability.

According to Dix, Finlay, Abowd, and Beale (2004:258), "designing for maximum usability is the goal of interactive systems." In this work, the principles to support usability are as follows:

Learnability: It should be easy for new users to learn how to use the system so that they can perform and interact effectively.

Flexibility: the various ways of exchanging information between the users and the system.

Robustness: Users should get a minimum level of support, which leads to effectiveness.

The evaluation of usability "is any analysis or empirical study of the usability of a prototype or system" (Rosson & Carroll, 2002:227). Rubin (2008:21) defines *usability testing* as "a process that employs people as testing participants who are representative of the target [audience to] evaluate the degree to which a product meets specific usability criteria".

The evaluation goals of usability testing are "to assess the extent and accessibility of the system's functionality, to assess users' experience of the interaction, and to identify any specific problems with the system" (Dix et al., 2004:319-320). It is expected that the usability evaluation provides feedback that can be used in different stages of the software development. Thus, there is a close connection between usability design and prototyping (Dix et al., 2004:319).

2.4.2 Usability Types and Techniques

In this project, one of the objectives is to design and develop a prototype, which also includes a user interface. "A crucial part of making a good user interface is to… test [user interfaces] for usability" (Lauesen, 2005:41). In order to conduct usability testing, it is necessary to use some usability types and techniques. The ones that are relevant to this project will now be explained briefly.

2.4.2.1 Laboratory Studies vs. Field Studies

There are two different types of usability evaluation: laboratory studies and field studies. This distinction is based on the environment in which the tests are conducted. The laboratory environment is a relatively "sterile" place in the sense that it is possible to control some of the features of the environment, for example the lights, noise, temperature etc. Field studies are held in the actual/original working environment, which "allows us to study the interaction as it occurs in actual use" (Dix et al., 2004:327-329).

The office environment and the workload of the conference manager have an obvious impact on the detection of fraudulent activities, since this task requires a certain amount of attention. That is why it is important to test the prototype in the working environment of the conference manager. Consequently, this project will employ the field study for the usability evaluation of the prototype.

2.4.2.2 Usability Techniques

There are a variety of evaluation techniques that can be employed to achieve usability goals (Dix et al., 2004: Chapter 9; Rosson & Carroll, 2002: Chapter 7). *Observation* is the main kind of evaluation in the evaluation process; data about the actual use of the prototype is collected by observing users interacting with it (Dix et al., 2004: 343). The focus should be on finding a set of observational techniques that best suit the requirements. *Interviewing* and *think aloud protocols* are the relevant evaluation techniques for this project:

Interviewing is a *query technique* that "relies on asking the user about the interface directly" (Dix et al., 2004:348). It gives an opportunity to gather the user's viewpoint. Dix et al. (2004:348) defines interviews as follows:

> An interview will usually follow a top-down approach, starting with a general question about a task and progressing to more leading questions (often of the form "why?" or "what if?") to elaborate the aspects of the user's response.

Planning is an important element of the interview; the core questions should be prepared in advance. It is also possible to adapt the form of the interview to each user since it is not a controlled experimental technique.

Think aloud protocol (TAP) is an observational technique that is based on the cooperation of the user. It is also a way to observe the interaction between the user and the system. Basically, "the user is asked to talk through what s/he is doing as s/he is being observed" (Dix et al., 2004:343). It is an easy-to-learn method, but its effectiveness depends on the choice of recording, e.g. paper and pencil, audio recording or video recording.

2.5 Summary

This chapter gives an overview of the theoretical, conceptual and methodological landscapes that inform this thesis. A target-oriented literature review enabled me to make conscious choices concerning the conceptual and methodological approaches. In brief, my choices as a result of the above discussion are:

1. A monitoring agent who is autonomous, reactive and proactive, developed based on its roles. Details of the development of the agent can be found in Chapter 3.

2. Agent-oriented methodology and iterative software development as the main software development approaches.

3. Usability testing as the evaluation method and a field study that includes observational techniques such as interviewing and think aloud protocols. The interview is planned to be semi-structured; that is, the answers of the participants are used in questions for detail. Evaluation and findings are presented in Chapter 4 and evaluation documents are included in Appendix 1.

3 Design and Development

Russell and Norvig (1995:33) state that "the notion of an agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents."[10] The nature of agency leads to some differences in practice that produce a distinction between the design and development of agents and objects or object-oriented systems.

In this chapter, the objective is to follow agent-oriented modelling and development practices in the creation of the monitoring agent.

3.1 The Agent-Oriented Approach

"Most recent software engineering methodologies are designed for an object oriented approach" (Caire et al., 2002). However, agents are different from objects, and although object-oriented methodologies are used to construct agent systems, there still exists an agent orientation in such efforts. Caire et al. (2002) claim that "agent orientation is … a paradigm for *analysis*, *design* and *system organization*" and add:

> …[Agent Oriented Software Engineering] AOSE can be divided into two broad categories. The first category aims to apply existing software engineering methodologies to AOSE…. The second category of work aims at developing a methodology from agent theory, mainly covering analysis and design…

The concept of *agent* lies at the centre of the design and development of the monitoring agent prototype, MONA.[11] First of all, an agent will be "one [that] acts" (Franklin & Graesser, 1997) by overtaking some parts of the Conference Manager's tasks in the workflow, which are related to the identification of possible fraud attempts.

A higher importance is placed on being *reactive* (Depke, Heckel, & Kuster, 2001) and *autonomous* (Depke et al., 2001) in the design process: MONA perceives its environment via GUI (communication with the conference manager), and MONA is designed to overtake some parts of the conference manager's workflow items; that is, MONA will be assigned tasks by the conference manager. The use of GUI can be considered to be a message passing

[10] The same statement is also mentioned in Franklin & Graesser (1997).

[11] I will refer to MONA as "she" from now on, simply because Mona is a female name.

from/to human and software agents, and MONA will perform the given task autonomously, i.e. without human intervention.

3.2 Agent Domain: Open Conference Systems

The conference management system[12] that will be used in this project is *Open Conference Systems* (OCS), which is developed as a part of the *Public Knowledge Project.*[13] It is a free web publishing tool that is provided for scholarly conferences with the GNU public licence, and is therefore used widely in different conferences,[14] which increases the use value of this research. OCS has many features, such as aiding the creation of a conference website, conference participant registration, online submission for abstracts and papers, a multiple-round review system and credit-card payment for registrations.

In this project, the emphasis is on the registration and payment processes of this web tool. The operational process is as follows: The attendee submits his or her own personal contact information. According to the given fee, the OCS prepares a bill and delivers this bill over to the online payment tool. By using a credit card or PayPal account, the attendee pays the fee. (The conference organization should establish a seller account in advance of using this tool in conference fee payments.) By default, an attendee who pays his or her fee may participate in the conference; however, the conference manager has a right to refuse any registration based on rules and regulations.

3.2.1. Current Situation in OCS

In OCS 2.0, there is no available check in the system to capture or gather information about fraud attempts.

The system comprises two aggregate reporting possibilities: *list of registrations* and the total *number of registrations*. The administrator can obtain a list of all registrations in an MS Excel file by using “Registrant Report”, which is stored in the “Reports” section under a

[12] There are various different Conference Management Systems on the market, such as EasyChair, OpenConf, EDAS Conference Manager, ConfTool, ConfMaster, IAPR Commence Conference System, COMS, Confious, Open Conference Systems etc. Some of them are commercial and others are open-source products. These have different features including registration, online submission, reviewing, payment support, data export and reporting.

[13] More information on this project can be obtained at http://pkp.sfu.ca/ (Last accessed 25th.November.2010).

[14] For a list of conferences where OCS has been used as a primary tool, see http://pkp.sfu.ca/ocs-conferences (Last accessed 25th.November.2010).

"Conference Site Management/Stats & Reports" link. However, detecting fraudulent registrations manually is not an easy task when there are many registrants.

In the registration/payment process, the `paypal_transactions` table is updated when a payment event is completed. One drawback is that there is no interface in the system that shows the stored payment events; the payment information exists but is hidden to the end user. In addition, the data stored on this table is not user-friendly. Every event is defined with a 17-digit alphanumerical payment reference. It is not always easy to compare the stored information with the information from the payment account manually because of these cumbersome payment references.

In short, this lack of reporting and investigation possibilities impedes the conference manager's work.

3.2.2 Finding Fraudulent Activities: User's Workflow Details

As a result of missing reports and functions, the conference manager[15] has to collect information about possible fraudulent activities manually. The workflow of the conference manager in identifying such potential fraudulent activities in the registration process is presented in the following paragraph as well as Figure 2.

The conference manager collects as much information about the registrants as possible in order to make a decision about accepting their registrations. In this process, one of the main tasks is to obtain a list of registrants who have already paid the conference fee. After receiving the lists of registrants and payments, the manager compares the items in these files by using the name, email address and country of the registrant, fee type, fee amount and date of registration. Registrants from some visa-required countries are checked additionally by using other kinds of information given at registration, such as country, affiliation, email address etc. Search engines such as google.com and pipl.com are other options for finding out more about the registrant and the registrant's affiliation. If there is still some suspicion about the registrant's intentions (whether the aim is to attend the conference or not), it is possible to ask for his or her CV to find more information about the registrant. After the conference manager accepts the registrations, he or she must update the records in the conference management system, and the process ends.

[15] The "conference manager" and "user" will be used interchangeably from now on.

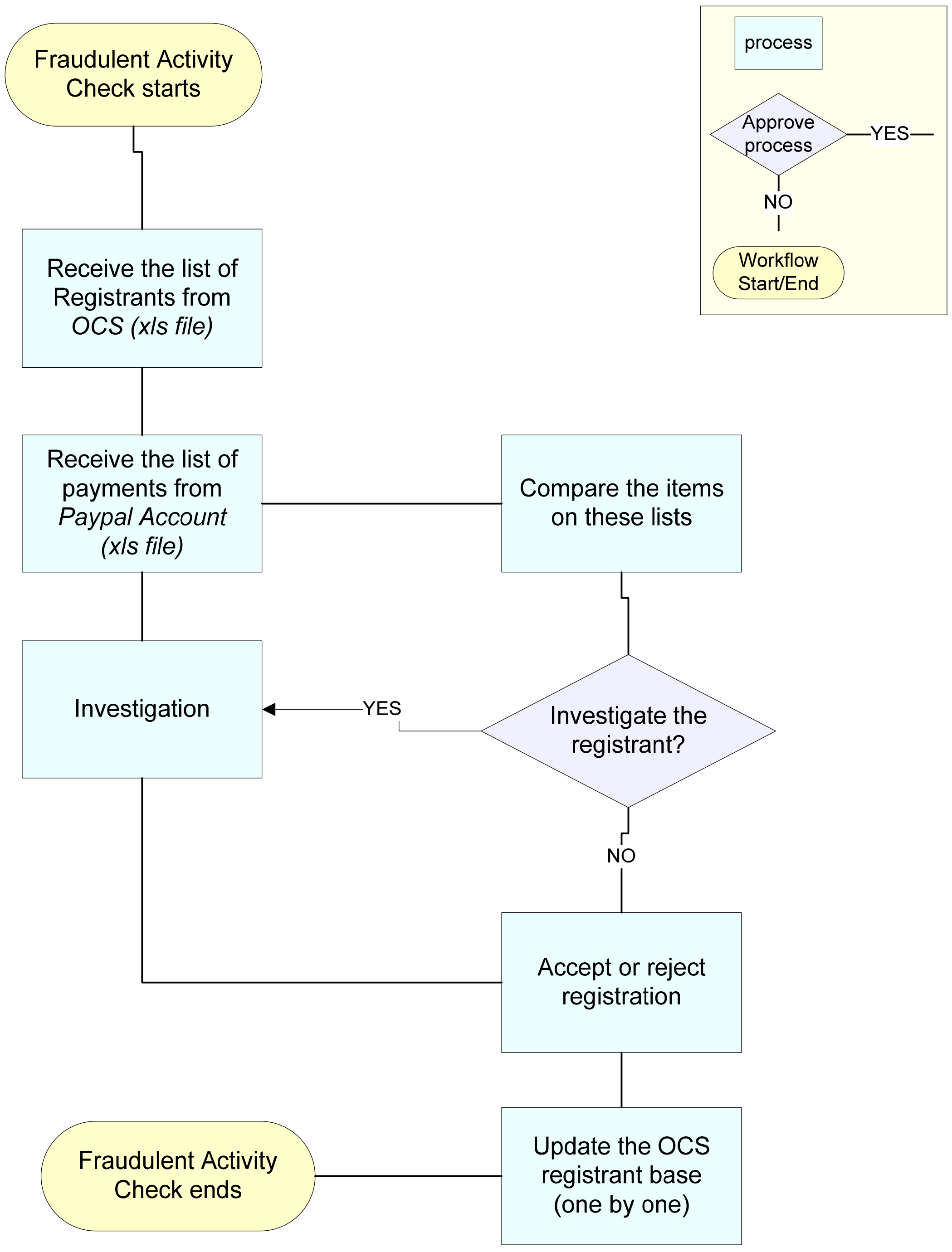

Fraudulent Activity Check starts
Receive the list of Registrants from *OCS (xls file)*
Receive the list of payments from *Paypal Account (xls file)*
Compare the items on these lists
Investigation
YES
Investigate the registrant?
NO
Accept or reject registration
Fraudulent Activity Check ends
Update the OCS registrant base (one by one)
process
Approve process
YES
NO
Workflow Start/End

Figure 2: **Finding Fraudulent Activities: Workflow Diagram of Conference Manager**

The leading disadvantage of this manual work is that it is highly time-consuming and open to human error, which might lower the quality of the output. Furthermore, in order to improve this process by getting *processed data* from the OCS environment and the payment system, it is necessary to have programming skills and this is not a common ability among conference managers.

3.3 Modelling the Monitoring Agent

The agent-oriented modelling process comprises the activities of "requirements specification, analysis and design" (Depke et al., 2001). The aim is to find out "the key abstractions of roles and responsibilities" (Bussmann et al., 2004: 90). Therefore, I have followed Depke et al. (2001) in the organisation of this section. In "Requirements Specification", a detailed picture of the monitoring agent system in terms of system functionality is given. In the "Analysis" part, the fraud indicators are analysed. Lastly, in "Design", the agent model is presented with its roles and tasks.

3.3.1 Requirements Specification

The purpose of this project is to develop an exception monitoring agent prototype to support conference managers in detecting fraud attempts in an OCS environment. In order to achieve this objective, the requirements have been carefully analysed. The system requirements specification is outlined below.

3.3.1.1 System Context of the Prototype

MONA, *Monitoring Agent for Open Conference Systems*, is a web-based automated monitoring agent system, which is integrated into the OCS environment.

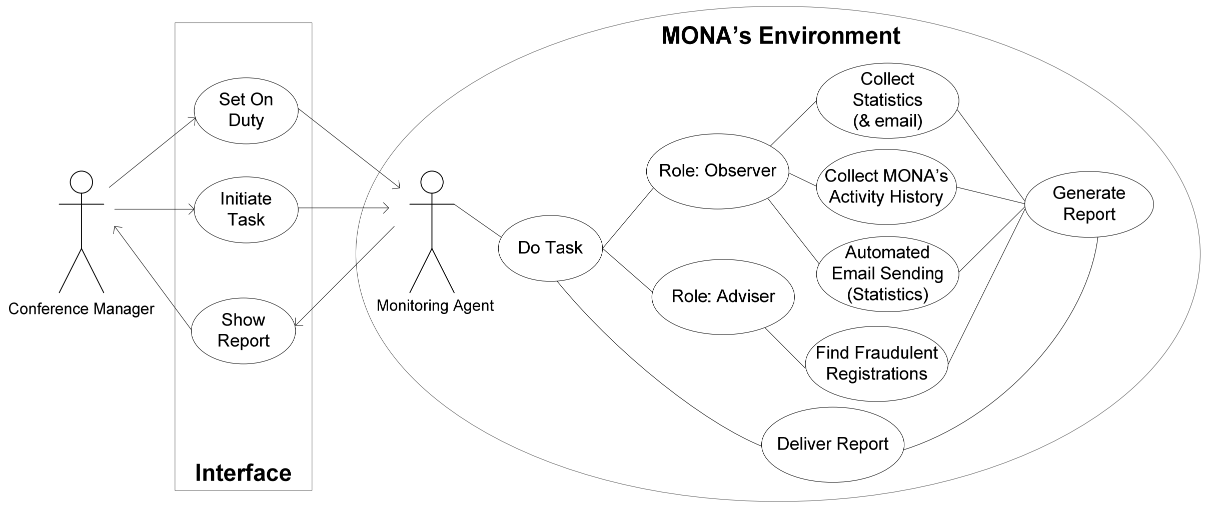


Figure 3: **Use Case Diagram for Monitoring Agent**
Source**: Adapted from Depke et al., 2001**

When MONA is *set on duty* by the user, it monitors the registration process of a particular conference. Its tasks include generating different kinds of reports in runtime, and presenting them either on the web or sending them via email. Figure 3, the use case diagram (which is adapted from Depke et al., 2001) gives an informal description of the functional and architectural requirements of MONA, where "Conference Manager" is presented as the main actor.

3.3.1.2 Major System Capabilities

This subsection lists functional and non-functional requirements. Functional requirements give a functional picture of the system, i.e. an overview of "how the system should react to particular inputs and how the system should behave in particular situations" (Sommerville, 2004:119). On the other hand, non-functional requirements are "the qualities" (Robertson & Robertson, 2006:10) and "the constraints on the services or functions offered by the system" (Sommerville, 2004:119). Table 1 and Table 2 (adapted from IEEE.Std.1233, 1998 Edition) list the functional and non-functional requirements, respectively.

3.3.1.3 Functional Requirements

***Table 1*: Functional Requirements**

No.	**Type**	**Functional Requirement**
1	Must	- MONA is a web-based monitoring agent application.
2	Must	- User sets MONA *On Duty* or *Off Duty.*
3	Must	- User configures MONA.
4	Must	**TASK 1:** - MONA presents a summary of registrations on the web. - Includes country list
4a	Must	- Connects to OCS database and obtains statistical information. - Includes country list.
4b	Must	- Generates a report and presents it on the web.
5	Must	**TASK 2:** - MONA presents a list of its own actions on the web.
5a	Must	- Connects to OCS database, and obtains its activity information.
5b	Must	- Generates a report and presents it on the web.
6	Must	**TASK 3:** - MONA presents a list of registrations based on payment transactions on the web. *Continued on the next page*

No.	Type	*continued* **Functional Requirement**
6a	Must	- Reads payments file and makes a list of payments.
6b	Must	- Connects to OCS database, reads `paypal_payments` table and makes a list of payments.
6c	Must	- Connects to OCS database, reads `users.registrations`, `user_sched_conf` tables and makes a list of registrants.
6d	Must	- Compares the lists of payments (the one from the database with the one from the payments file). - The records that satisfy the comparison condition should be kept in a separate list.
6e	Must	- Compares the database-list with the payment-file-list. - Marks the items in the list with: full match, missing name/email, person does not exist.
6f	Must	- Produces a report that includes the findings.
6g	Can	- This report might include further investigation possibilities: Pipl/Google search for participants name and/or institute and/or country.
7	Must	**TASK 4:** - MONA presents a short summary of registrations and sends them to the user by email upon request.
8	Must	**TASK 5:** - MONA presents a short summary of registrations and sends them to the user by email every day at 08:00.
8a	Must	- Connects to OCS database, and obtains statistical information.
8b	Must	- Presents statistical information in an email which is sent automatically every day at 08:00.

3.3.1.4 Non-functional Requirements

***Table 2*: Non-Functional Requirements**

No.	Non-functional Requirement	Description
1	Accessibility	Accessible to the user
2	Availability	Available as long as the partner systems (conference management and payment) are available.
3	Dependence on other parties	Depends upon the information that is going to come from OCS and payment systems.
4	Efficiency (resource consumption for given load)	OCS should continue to work without having performance problems while MONA is set on duty.
6	Extensibility	Not necessary to have an "application programming interface".
7	Performance and response time	Produces the result report as soon as possible. Report producing time more than 3 seconds is not affordable. *Continued on the next page*

No.	Non-functional Requirement	*continued* Description
8	Maintainability	Easy to maintain.
9	Open Source	Not a commercial program, freely available to the public.
10	Quality	Delivers a high quality result—reliable.
11	Robustness	Program should not get runtime errors during the intended execution of the program.
12	Stability	Program should deal with errors so that it will not crash during the execution of the program.
13	Visualisation and usability	User-friendly GUI. Produces easy-to-use result-report.

3.3.1.5 User Characteristics

Single User: Conference Manager. The user should have a valid id and password in the OCS environment, and this id should have the role of administrator in the conference system.

3.3.1.6 Operational Scenario

- User logs on to MONA's website.
- User configures MONA and sets it on duty. As soon as the user has done so, it will send automated emails that include a short summary of registrations every day at 08:00.
- User can assign different tasks to MONA. It presents different reports according to the chosen tasks.
- User can set MONA off duty. It stops sending automated emails, if it has been configured to do it, and cannot be assigned to tasks.

3.3.1.7 Prototype Environment

Everything needs an environment which to exist; "[a]n environment provides the conditions under which an entity (agent or object) exists" (FIPA, 2003). I will now describe the different parts of MONA's environment.

OCS System Operating Environment:

1. Server Environment

The server where OCS is located uses the following programs for OCS: PHP 5.1.6*,* MySQL 5.0.45, Apache 2.2, and the operating system Red Hat Enterprise Linux.

2. Payment Plug-in

The OCS uses PayPal Plug-in, which enables users “to use all major credit cards ... when paying through the system.”[16] To use this plug-in, it is necessary to get an account from PayPal via a web-based interface in which customers can log on and monitor their accounts. Here, there is the opportunity to obtain the history of an account as a text file. This file will be used as an input by the prototype.

Agent Environment:[17]

The quotes in this section are taken from Weiss (1999, cited in FIPA, 2003) and Russell (1995, cited in FIPA, 2003). The basic characteristics of MONA’s environment will include the following:

“Accessibility—To what extent is the environment known and available to the agent?” The monitoring agent has access to the OCS database and to the file space. In the database, its rights are limited to reading and updating payment dates only, and it can read the files under the “uploaded” map.

“Determinism—To what extent can the agent predict events in the environment?” The agent predicts the possible fraudulent registrations.

“Diversity—How homogeneous or heterogeneous are the entities in the environment?” The OCS database is a relational database, and MONA has access to several tables in this RDBMS. MONA also reads the PayPal account summary file, which is located in the OCS file space under the “uploaded” map.

[16] From the actual OCS, http://www.ocs.uib.no/thesis/ocs/ (Last accessed 25th.November.2010). The information is under Home > Open Conference Systems > User > Conference Site Management > Plugin Management on this webpage.

[17] Adapted from FIPA (2003).

"Controllability—To what extent can the agent modify its environment?" MONA modifies its own tables, `agent` and `agent_events`, in the OCS database.

"Volatility—How much can the environment change while the agent is deliberating?" Until the registration deadline is reached, there might/will be some changes in the OCS database (attendees will register for the conference) while the monitoring agent is on.

"Temporality—Is time divided in a clearly defined manner?" The tasks are defined clearly. The actions happen synchronously.

"Locality—Does the agent have a distinct location in the environment which may or may not be the same as the location of other agents sharing the same environment?" This prototype works with one agent, and the monitoring agent is located in the same place as the OCS environment.

3.3.1.8 Prototype Architecture

MONA is designed as a reactive system[18] in which the data is processed. It processes the stored data in the OCS database and the payment file according to the inputs from the monitor (the chosen task). It completes its tasks and generates reports which are presented on the web and/or sent via email.

3.3.1.9 Documentation Requirements

The program is supported with information about MONA and its tasks. Completed tasks include information about MONA's header information. Coding is supported with JavaDoc-type documentation.

3.3.1.10 Programmatic Requirements

The Java programming language is chosen as the programming language to develop a desktop application. Since this programming language can be used on different platforms, it is an advantage to use Java on this project. Eclipse Ganymede is used as the programming tool. Eclipse Ganymede has also good tutorial coverage on its website. Moreover, my BA included

[18] Lee (2000) gives an overview of reactive-system approaches to agent architectures. He categorises reactive systems based on their tasks and environment properties. According to this categorisation, MONA can be seen as an "Automated Monitoring and Controlling System".

two courses on Java programming: an introduction to programming and a specialisation course in Java programming.

On the server side, the latest version of MySQL (version 5.0) is in use. The PC I am using to develop this project runs on Windows Vista. The driver (v. 5.0 and v. 5.1) which will support the connection between the MySQL database and the desktop application is not mature yet. This might create some problems during development.

The second programming language choice is PHP. Programming with PHP is similar to Java, which is an advantage. Moreover, php.net has good documentation available on the Internet. The tool needed to program with PHP is a source code editor. For this, Eclipse Ganymede with PHP plug-ins[19] is used.

For server file operations, the chosen tool is SSH Secure File Transfer Client (3.2.9) which will be used in the installation process for OCS, as well.

For database operations, phpMyAdmin is the main tool. It "is a free software tool written in PHP intended to handle the administration of MySQL over the World Wide Web".[20] An overview of the programs used in the development process is given below in Table 3.

***Table 3*: Programs/Programming Tools**

Programs/Programming Tools	Description
Eclipse Ganymede	Eclipse development platform for Java Source Code
Eclipse Ganymede/PDT	Eclipse development platform for PHP
SSH Secure File Transfer Client	Secure transfer of the files; installation of OCS
phpMyAdmin	Administration of the OCS database

3.3.1.11 Other Requirements

The program will be delivered with a GNU license (non-commercial, open source).

[19] More information can be found at http://www.eclipse.org/pdt/ (Last accessed 25th.November.2010).

[20] http://www.phpmyadmin.net/home_page/index.php (Last accessed 25th.November.2010).

3.4 Analysis

The process of analysis is intended to show what the agent system does. The structure of MONA is presented in Figure 4 in an agent class diagram with "generic agent notation" (FIPA, 2000), and in Figure 5 it is presented in a structural model which is a "class diagram" (Depke et al., 2001).

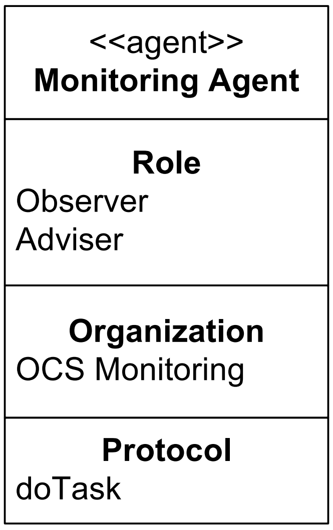


Figure 4: **Agent Class Diagram**
Source: **Adapted from FIPA (2003)**

In the diagram in Figure 5, there are two main files which hold the agent system classes: agent.php and agentUtils.php. The history of the payment account, which presents the information about payments, is placed in the payments.txt file in the upload directory. The DB_read class is responsible for communication with the OCS database, in which tables that are used in registration process and the ones that are used by the agent are stored.

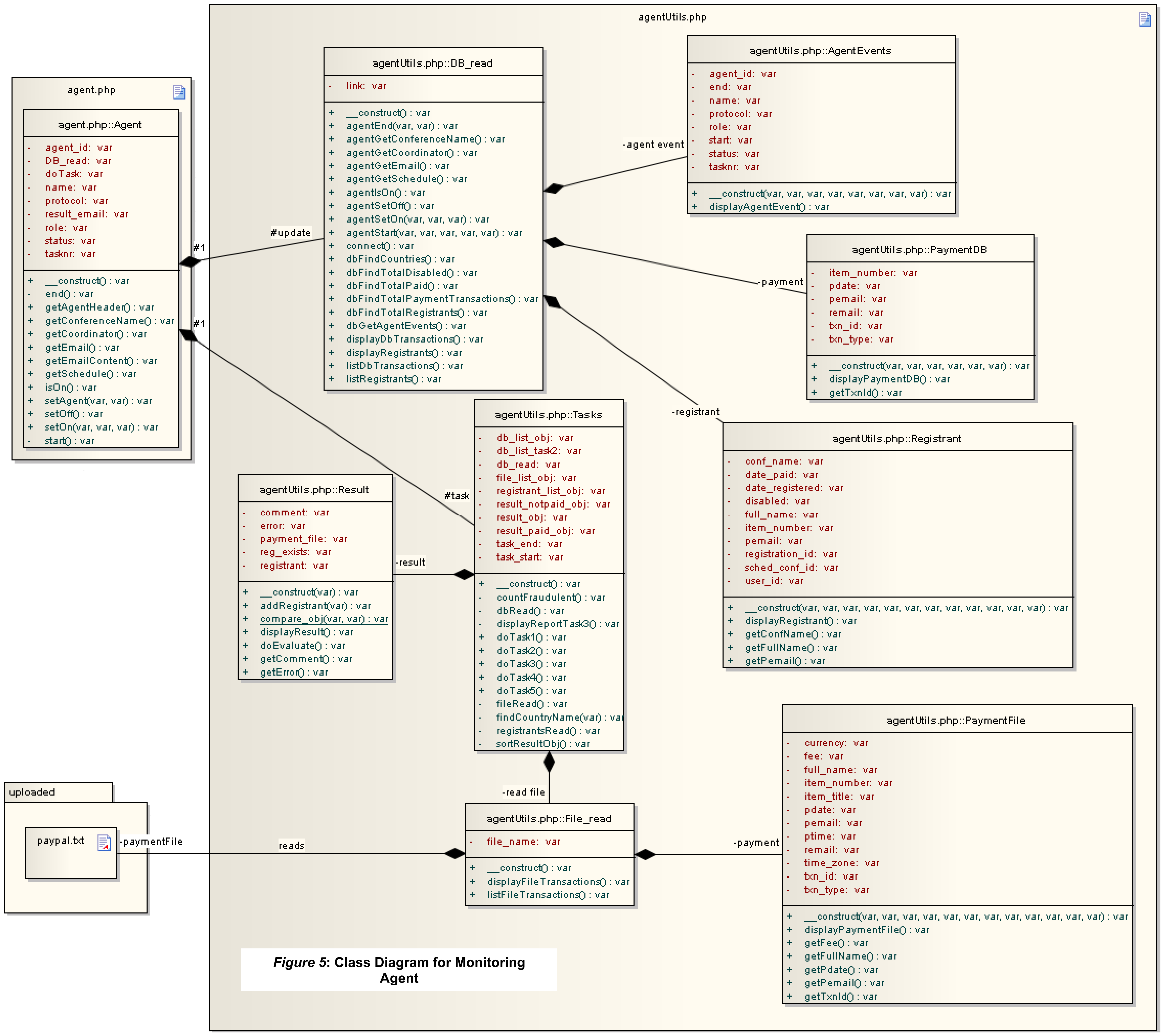


***Figure 5*: Class Diagram for Monitoring Agent**

3.4.1 Definition of Fraud

Fraudulent activity is defined as a registrant's using the conference as a way of entering the country rather than attending an academic event. It is very hard to evaluate the registrant's intentions; however, the stored data in OCS gives us some details, which makes a kind of evaluation possible.

An analysis of the conference manager's workflow for finding fraud attempts shows that it is necessary to compare the data that is stored in the payment account with the data that is stored in the Conference Management System (CMS). The result of this comparison allows for an opinion about whether further investigation is necessary. For example, if the payer is different than the actual attendee, it is a potential warning signal.

MONA classifies the intentions of the registrants based on the information given in the registration process. The fraud indicators that are used in this project are the registrant's payment data, full name and email address.

There are also certain assumptions here, as the interviews with experienced conference managers have made clear. These assumptions support the interpretation of stored data in the CMS, and they are implemented in the prototype. They can be listed as follows:

- Payment is personal; every registrant has the intention to pay only for him- or herself and not on behalf of somebody else.
- The registrant uses the same full name and email address both in the registration process and the online payment process.
- There will be few exceptions to the aforementioned assumptions. In such cases, the registrants will need to get in touch with the conference manager so that it will be relatively easier to deal with these cases.
- The conference manager will use internet search engines such as pipl.com and/or google.com to gather information about the registrant in a possible case of fraud by searching with the full name and country information.

There are also some assumptions related to the CMS and use of payment account:

- There is only one conference defined in the system.

- The payment account, in which the conference fees are collected, is/can be used for several purposes, e.g. to collect fees for several conferences or other types of payments; i.e. it includes both the relevant and non-relevant payment records.

3.5 Design

The process of design "concentrates on the question of how the system will function" (Depke et al., 2001). I will now explain briefly how MONA will work as a monitoring and single user's agent, before describing MONA's roles.

3.5.1 Monitoring Agent

The ways in which the system functions should satisfy the criteria for being an agent: autonomy, reactivity and proactivity.

First of all, MONA should fulfil its tasks autonomously, i.e. without human intervention. Nevertheless, autonomy is not the only feature an agent should have. It should also be "capable of flexible autonomous action in order to meet [its] design objectives" (Jennings & Wooldridge, 1998:4). This can be achieved by being reactive, responding to environment in a timely fashion; it perceives its environment and responds to any changes. On the one hand, it responds to the user via the interface: MONA is configured, gets tasks from the user, communicates with the OCS database and payment file and collects necessary information to create and deliver reports. However, MONA does not have to react to the changes in the OCS environment in real-time, since its role is that of an assistant which contributes whenever needed.

Another important feature MONA should have is proactivity. It is important to mention that this feature has to be shaped according to the limits of the technology used to create the agent. PHP, the main programming language, does not support the necessary tools/functions as event handlers that work in line with the MySQL database. As a result of this disadvantage, I design MONA's (limited) proactivity feature as follows. Using the resources of the server, I set a crontab-job on the server so that it will work at 08:00 every day. MONA checks the registration data, prepares an overall report and emails it to the user. The user has the option of using/accepting this feature. If the user does not want it, he or she can configure MONA to turn this feature off.

3.5.2 Single User Agent:[21]

MONA is designed to automate some parts of the conference manager's tasks primarily in order to detect possible fraudulent activities in the registration process. MONA will work as a single *user* agent,[22] as a personal assistant to the conference manager (human agent). Collaboration between these two agents will occur in order to complete the task of detecting fraud attempts in the OCS environment.

MONA will assist the user by supplying necessary information and possibilities for investigation. In other words, MONA's purpose is not that of taking decisions but supplying necessary information to the user about registrant intentions based on the above-mentioned specific rules and assumptions.

MONA will also act as an *interface* between the OCS and the user. The user can set MONA on duty, upon which it will do the chosen tasks immediately. It collects any available information about the registration process, such as the conference, registrants, payments, fees, OCS registrants, and itself. It then evaluates and reformulates this information in different ways in order to assist the user's responsibilities.

3.5.3 MONA's Roles[23]

MONA is designed to have two distinct roles: It is both an *observer* and an *adviser* in detecting possible fraudulent activities. In Figure 6, adapted from FIPA (2000) and Caire et al. (2002), these roles are presented as an agent class diagram with "capability description notation" (FIPA, 2000).

Observer Role:

MONA presents the stored information in the OCS database so that the user can follow the registration process easily. This is shown as an agent class diagram in Figure 6. The goals MONA wants to achieve are as follows:

---

[21] Lashkari, Metral, and Maes (1998) describe a *single user email agent* in their work.

[22] In Jennings and Wooldridge (1998:5), such kinds of agent systems are called "expert assistants".

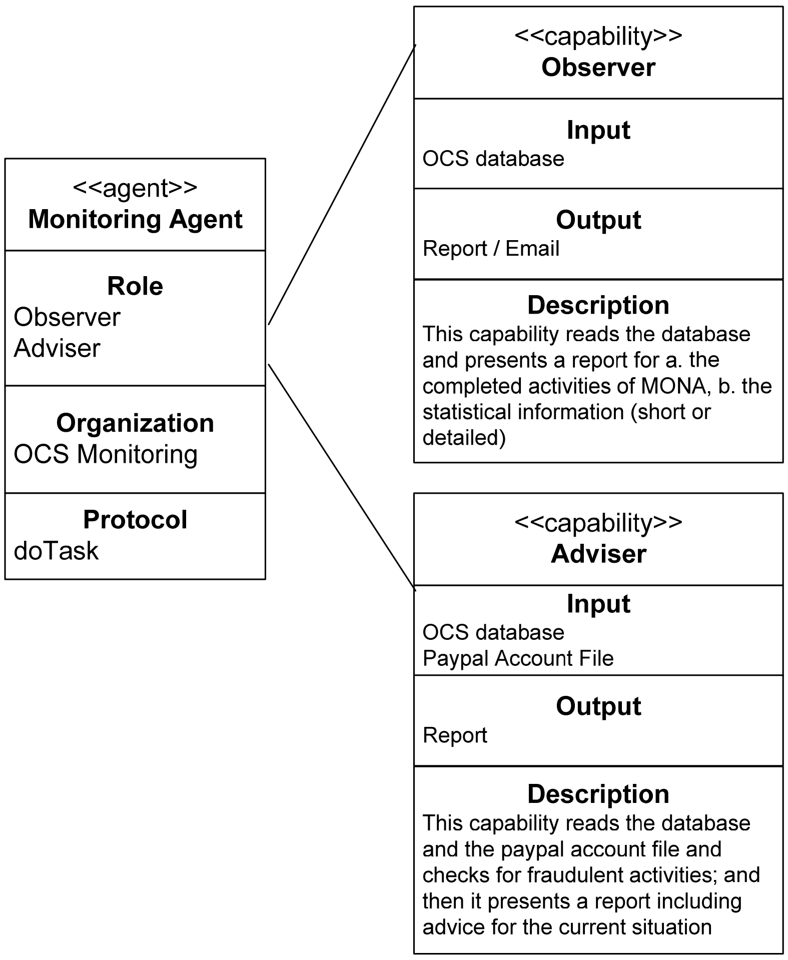
[23] See also Chen, Dolonen, and Wasson (2003) and Chen and Wasson (2002) for "monitoring agent" implementations. This section is adapted from these sources.

Statistical information about the registration process: How many registrants are there in the database? How many of them have a status of accepted, pending or rejected? How many registrants have or have not paid their fees? (This function is implemented in Task 1 and Task 3 in Table 1.)

Historical information about MONA's actions: Which of the tasks has it completed, and when? (This function is implemented in Task 2.)

Advisor Role

MONA monitors the registration process and uncovers the possible fraud attempts, generating a report that includes the evaluation results. The objective is to allow the user to maintain broader information about the registration process. MONA's goal is to encourage and support the user's awareness by doing necessary routine checks, and supplying necessary and adequate information. (This function is implemented in Task 3.)



<<agent>>
Monitoring Agent
Role
Observer
Adviser
Organization
OCS Monitoring
Protocol
doTask
<<capability>>
Observer
Input
OCS database
Output
Report / Email
Description
This capability reads the database and presents a report for a. the completed activities of MONA, b. the statistical information (short or detailed)
<<capability>>
Adviser
Input
OCS database
Paypal Account File
Output
Report
Description
This capability reads the database and the paypal account file and checks for fraudulent activities; and then it presents a report including advice for the current situation

***Figure 6:* Agent Class Diagram with Capability Description Notation**

***Source*: Adapted from FIPA (2003)**

3.5.4 MONA's Tasks

MONA's basic goal is to collect necessary information from the OCS environment (registrations and payments) and from payment data (payment file), process the information, and present the results in a report to the user in a timely fashion. A brief description of MONA's tasks categorised according to its roles is given below:

3.5.4.1 Operation Details for Detecting Possible Fraud Attempts

As an adviser, MONA detects the matching payments in the database and in the file by using the *transaction id*[24] which is an alphanumeric 17-digit reference created at the actual time of the payment. After finding the records of registrants who have not paid their fees, MONA checks whether the name and email address of the payer matches that of the registrant in the system, and flags the record in the following cases:

Full Match: The payer and the registrant are the same person. This record is labelled as "Accept!".

Not Matching Name/Email: The payment information matches, but the name/email of the payer is different than that of the registrant. This record is labelled "Check!" with a comment showing the data that does not match.

Unfound Registrant: The payment information matches, but neither the email nor the name of the payer is found in the database. This record is labelled "Fraud?" with the comment "Missing record in OCS".

Lastly, MONA presents the findings in a user-friendly report. This report includes a link with further possibilities for investigation, such as the information in the registrants base and payment file, and a link that brings up a browser window where the full name of the registrant has been searched for at google.com (Task 3).

3.5.4.2 Other Tasks

Other tasks are defined for MONA as an observer so that it can assist the user by supplying different kinds of information about the registration process.

[24] `Transaction id` is stored in the `Paypal_Transactions` table in the OCS database. However, none of the available reports in the OCS environment presents this very important piece of information. The only way to reach this information is to have the skills to use a database management system or any tool that searches the MySQL database, such as phpMyAdmin.

These tasks are:

1. User sets MONA on duty or off duty.
2. User can configure MONA to send an automatic email to the user every day at 08:00 while it is on duty. This task ends as soon as MONA is set to off duty (Task 5).
3. Mona can be assigned to perform the following tasks:
    a) Presenting a summary of registrations (Task 1),
    b) Presenting a list of its actions (Task 2),
    c) Presenting a list of registrations based on payment transactions and fraud check (Task 3),
    d) Sending an email to the user with a short summary of registrations (Task 4).

3.5.6 DoTask Protocol

Depke et al. (2001) define protocols as follows:

> In the analysis of the scenarios from requirements specification protocols can be derived that agent roles have to execute. The interaction of pairs of roles is examined in order to find protocol steps and protocol alternatives. Basic interactions between two roles are projections of more complex interactions… A set of graph transformation rules forms a protocol. It should be possible to derive all scenarios from requirements specification by the application of rules of a protocol.

MONA has two distinct roles which do not overlap. Thus, the level of complexity is low. It is possible to create a protocol for each task, but as a result of the low complexity, I prefer using just one protocol, *doTask*, where MONA is assigned to do a monitoring task on behalf of the user.

Roles can be represented in statecharts, which show the different states a role has to accomplish. The statecharts (adapted from Depke et al. [2001]) for MONA's different roles are presented below in Figure 7.

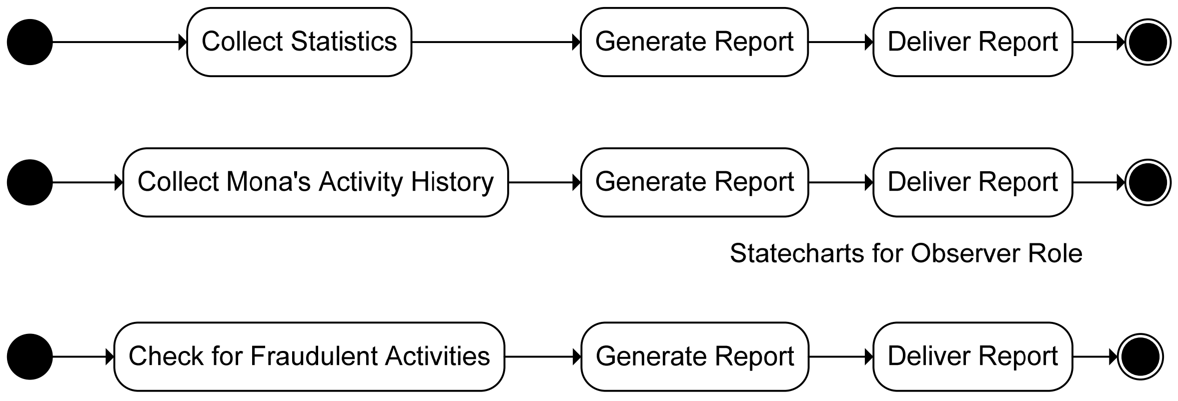


***Figure 7*: Statecharts for Different Roles of MONA**

***Source*: Adapted from Depke et al. (2001)**

3.5.7 Challenges Related to Designing Agents

In the modelling and construction of agents, Meas (1997) mentions two important problems related to competence and trust:

1. The *competence problem* has to do with how the agent can be competent enough to achieve a task; as Meas puts it, "How does an agent acquire the knowledge it needs to decide when to help the user, what to help the user with, and how to help the user?"

In this project, the abilities of the agent are explicitly defined. The rules which will be used in fraud detection and evaluation come directly from the conference manager's workflow for this special task. Secondly, when modelling MONA, it is necessary to set a hierarchy between the software agent (MONA) and human agent (conference manager) as a result of the domain. The conference manager (user) has a higher rank in the hierarchy than the monitoring agent, and the role of the user has a broader responsibility in the organisation of the conference. It should be the human agent who starts up the monitoring agent by using the agent interface. Thirdly, the scope of "help" is clearly defined in the modelling process: the output will be a report detailing the possible fraud based on the stored data. This report will be processed by the user, since the user is the one responsible in the organisation for such cases.

2. The *trust problem*, in Meas' words, is associated with the question of "How can we guarantee that the user feels comfortable delegating tasks to an agent?"

This question is especially important for this project, since "the degree to which people are willing to delegate duties to monitoring agents depends on how much they trust the capabilities of that agent" (Hayes, 1999).

Here, the hypothesis is that if it is possible to show what MONA does exactly what it is asked to do and that it does not make mistakes, the user will be convinced to use the result report created by the monitoring agent. This is related to the research question I mentioned previously: Is the outcome of collaboration with the monitoring agent reliable? Is it convincing enough for further collaboration? Some aspects of this hypothesis are addressed in Chapter 4.

3.5.8 Collaboration between MONA and the User

Considering the agent domain and the fact that MONA is a prototype, it is necessary to convince the user that Mona will be both capable of executing tasks and secure enough so that he or she can delegate some of his or her tasks to MONA. These issues are related to the above-mentioned challenges of competence and trust.

I believe that it is possible to overcome these challenges by carrying MONA's feature of autonomy to a different level, which is referred to as "adjustable autonomy" (Scerri, Pynadath, & Tambe, 2003 cited in Wooldridge, 2009: 23; Maheswaran, Tambe, Varakantham, & Myers, 2004). According to this idea,

> … the control of decision-making is transferred from agent to a person wherever certain conditions are met, for example:
>
> - when the agent believes that the human will make a decision with a substantially higher benefit
> - when there is a degree of uncertainty about the environment
> - when the decision might cause harm, or
> - when the agent lacks the capability to make a decision itself. (Scerri, Pynadath, & Tambe, 2003, cited in Wooldridge, 2009:23).

As a result of the nature of the domain, the conference manager is/should be the person who is responsible for taking actions in fraud situations. Secondly, it is not possible for a monitoring agent to gather all the information (e.g. CVs) necessary to decide whether a registrant is committing fraud. Thirdly, if Mona decides which activities are fraudulent instead of which activities might be fraudulent, a lack of experience on the part of the user might lead to harmful decisions for the organisation.

Therefore, MONA will assist the Conference Manager:[25]

- By sharing the complexity of possible fraud detection tasks with the conference manager
- By performing search and evaluation based on specific rules on behalf of the conference manager
- By giving hints about exceptions (fraud) that might occur in the workflow, in which the conference manager is the one responsible for detecting such situations
- By helping different users (for example, the conference manager and the financial officer) collaborate in fraud detection. In such cases, the information supplied by the agent will also be useful in creating a common language which will be used to express information about the fraud. This will be achieved by both the performance of the program and the result report.
- By monitoring exceptional events/activities that happen in a workflow.

In Figure 8, the interaction between the user and MONA is presented in a sequence diagram, which is adapted from Odell, Parunak, and Bauer (2000). Here, conference manager is the main actor and communication between him or her and MONA happens with the help of an interface. When the user assigns tasks to MONA (task specification), MONA automatically changes its roles from one to another according to the chosen tasks. After it has completed the task, it delivers a result report to the user via GUI.

[25] Adapted from Maes (1997).

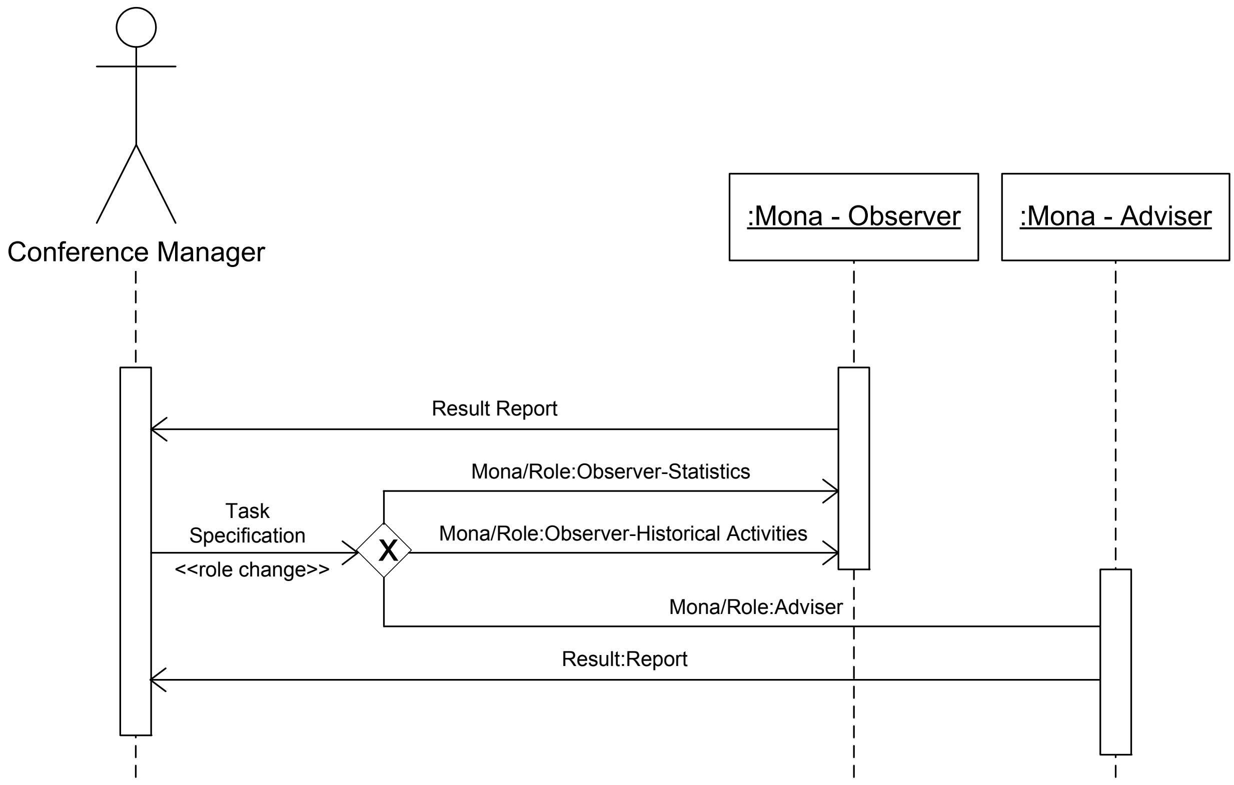


***Figure 8*: Sequence Diagram**
***Source*: Adapted from Odell et al. (2000)**

3.6 Integration of MONA with OCS Environment

3.6.1 Architecture[26]

MONA is integrated into the OCS environment where the agent application is also located.

In Figure 9 (adapted from Chen and Wasson [2002]), the integration of the monitoring agent with the OCS environment is shown. Regarding the database, MONA has access to the OCS database and the file space, which are located on the same server. MONA's own application tables are stored in the OCS database.

[26] This section is adapted from Chen and Wasson (2002).

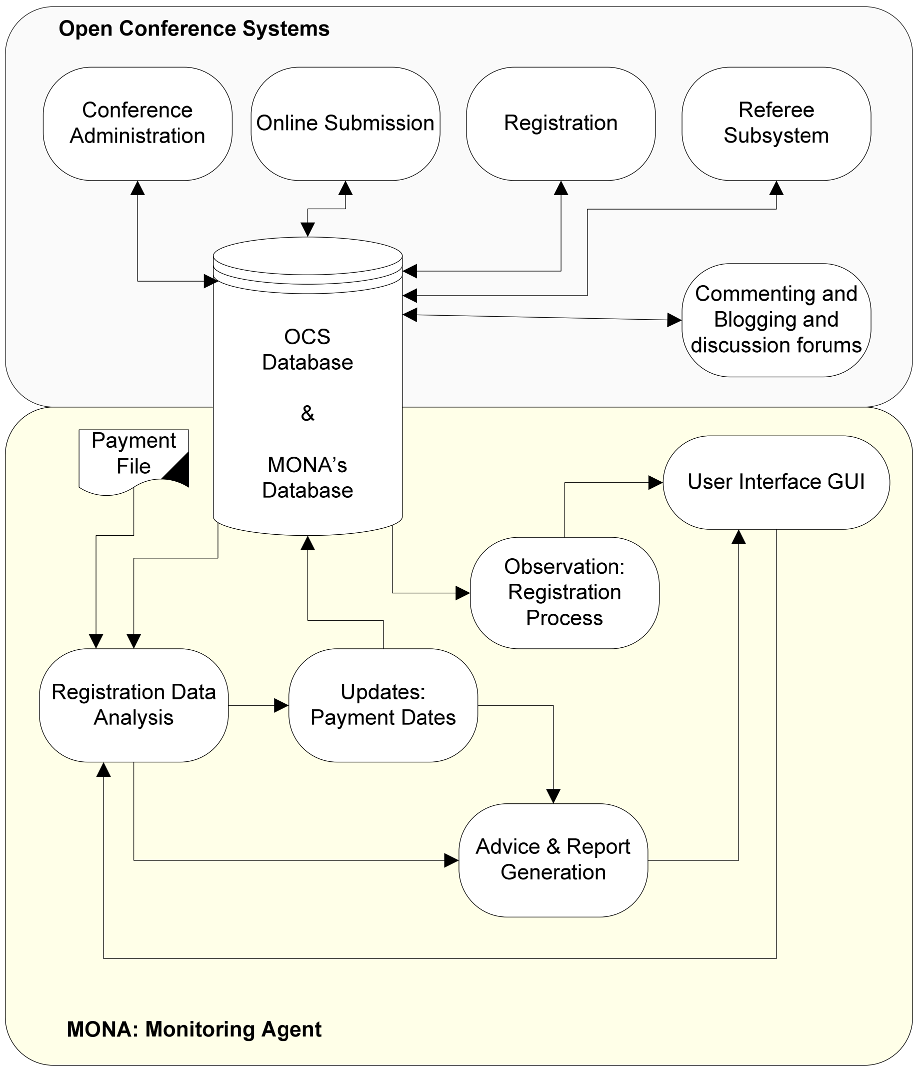

Open Conference Systems
Conference Administration
Online Submission
Registration
Referee Subsystem
Commenting and Blogging and discussion forums
OCS Database & MONA's Database
Payment File
User Interface GUI
Observation: Registration Process
Registration Data Analysis
Updates: Payment Dates
Advice & Report Generation
MONA: Monitoring Agent

***Figure 9*: Integration of MONA with the OCS environment**

***Source*: Adapted from Chen and Wasson (2002)**

3.7 Implementation Details: Iterations

The implementation process is organised in different iterations according to the principles of iterative and evolutionary development, which was outlined in Chapter 2. The aim here is to present a working and release-ready version of the prototype after each iteration.

3.7.1 Iteration 1

In this first iteration, I began coding with Java and used Eclipse Ganymede as the programming tool.

This first version of the prototype is considered to be a desktop application. The reason behind choosing a desktop application is that working with Java presents an opportunity to evaluate the functional objectives very quickly; in addition, the server configuration does not support some other kinds of development, such as C# and Java servlets.

Another important point in this iteration is to thoroughly learn how the payment account works, what kind of development possibilities it has, and how I can create test data for my project.

3.7.1.1 Test Driven Development

I have followed the basic steps of *test-driven development* (TDD) (Ambler, 2002-2009). This type of development is a combination of the *test first* principle and *refactoring*. In TDD, the test code is written first, and then the functional code is written. Running the test code is sufficient to determine whether the code fails. This can be done with a test suite, or subsets of the test suite can be run separately. If the tests fail, then it is obvious that changes in the functional code are necessary. Otherwise, it passes, and the next step in development can be taken up.

Following TDD gives me a chance to follow the working parts of the prototype, which indirectly helps me to focus on the parts of the prototype that are not working and not implemented. An advantage of using TDD is that eclipse.org has good documentation on how to do TDD, which both helps me and shortens the development process, as it enables granularity in the development process. The *Unit Test* extension, which supports the above-mentioned details, is used in this process.

3.7.1.2 Challenges

It is necessary to use a driver to connect to a remote database on the Internet. I used two different versions of this driver: MySQL Connector/J 5.0 and MySQL Connector/J 5.1. However, when I tried this, I got error messages and it was not possible to connect to the remote OCS database.[27] Moreover, a desktop application was not a good choice, even though this prototype would serve just one use: conference management. This implies a disadvantage: If the conference manager wants to present details by using MONA, having the

[27] I did some research on this problem and found that some other researchers have had the same problem. I tried the solutions they mentioned, but these did not help. I believe that the problem was related to the network so that the *connect timeout* of the database was not enough to establish a healthy connection.

application on a (stationary) computer might be a burden in some cases. The prototype should preferably have greater flexibility in its availability, for example, in a web-based application. As a result, it was necessary to change the application's implementation route to a web application.

3.7.2 Iteration 2

In this iteration, web-based prototype is implemented by using PHP.[28] This choice helped in overcoming the above-mentioned challenges.

I choose to work on the most important functions, such as reading the OCS database for payments and registrations, reading the payment file, comparing them and presenting the result on a webpage. All these functions are tested both separately and together (functional and system testing). Test cases are designed based on requirements. This is called "requirement-based testing" (Sommerville, 2004:552). According to the findings, the prototype is refactored.

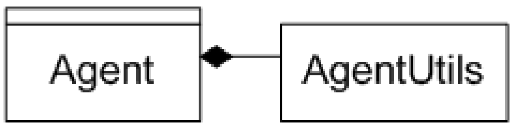


***Figure 10*: MonaUtils is a part of MONA**
***Source*: Bauer (2002)**

The object classes are combined under the `AgentUtils` class so that they will be used only by the `Agent` class. This is shown in Figure 10 (adapted from Bauer [2002]).

3.7.3 Iteration 3

I focus on the usability details in this iteration. It is important that the output should not only give the precise result, but also be understandable and look elegant.

Some other features were added in order to reach this objective, such as some new functions and a redesign of the webpage and MONA's picture shown in the webpage.

The content of the package Mona (scripts) and how it interacts with the OCS environment is shown in Figure 11. As an advantage of web-based programming, as soon as MONA's system files are copied to the server, MONA can be considered to be "live". However, in order to assign it "monitoring tasks", MONA has to be set *On Duty*. The user manual of MONA is included in Appendix 2.

[28] While writing code with PHP, I used mostly internet resources, especially www.php.net (Last accessed 25th.November.2010) and Vaswani (2005).

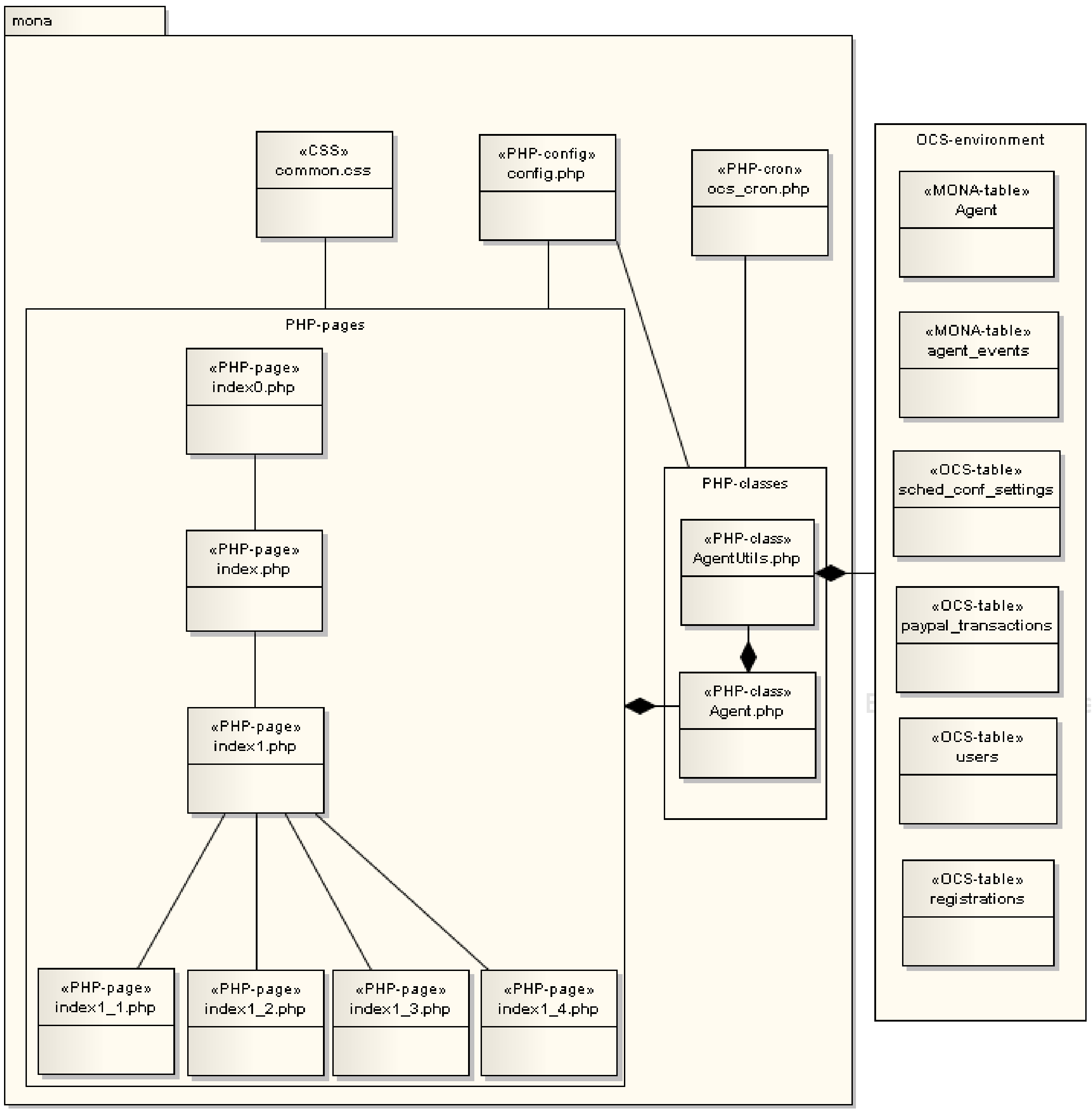


***Figure 11*: Content of package Mona (scripts)**

4 Evaluation and Findings

The main goals of the evaluation are to assess the usability of the prototype and to answer the research questions.

4.1 Usability Testing

The theoretical background for usability testing is covered in Chapter 2. In the following, the process of usability testing will be presented at the operational level.

4.1.1 Test Participants

The test user group that is best suited for the usability testing in this project is the actual users of this prototype. Seven participants, who represent "the population the system [OCS and the prototype] is designed to support" (Rosson & Carroll, 2002:254), are chosen to perform the usability testing. Their characteristics can be listed as follows:

- They have experience in conference participation and/or as a conference manager; two of them have also experience in using OCS.
- They are in the age group 30–55.
- They have higher education (BA or MA degrees in social sciences).
- They form a multicultural user group (Norwegian, Spanish, Sudanese and Turkish).
- Five of them are female and two of them are male.
- None of the participants has prior personal experience working with an agent system.

It is important to note that the selection method I apply here is not "sample representativeness" in a statistical sense, but rather a target-oriented selection of test participants in a qualitative sense (Ragin, 1994: 113). By selecting and comparing a small number of people who are different from each other in age, education level, cultural background and professional experience, I intentionally and systematically increase the probability of detecting usability errors and thus increase the "reliability" (Ragin, 1994: 142-3) of the usability test. This selection helps decreasing the probability of randomness in the answers given by the participants in the interviews. Moreover, the group's small number gives me the chance to do in-depth interviews and analyses.

4.1.2 Test Plan

The test is organized in two parts. The first part begins with the researcher's verbal and written dissemination of information about the prototype and the test process, and the participants are asked to complete a set of tasks. In the second part, I conduct an interview session to collect data about both the usability of the prototype and the research questions (see Chapter 1).

4.1.2.1 Context

The test is held during working hours at the workplace (office) of the participants in order to observe the use of the prototype in their natural working environment. While the test is conducted, the participants are left alone to fulfil the tasks; they receive no help from their colleagues.

4.1.2.2 Data Collection Methods

In the first part of the test, a short survey is used to collect data about the participants. While they are working with the various tasks, they are also asked to think aloud. The participants are being observed in this process, and notes are taken. In the second part of the test, a semi-structured interview technique is the major data collection method. The interview questions are prepared in advance[29] and these questions are used to guide the interview process. As much as possible, the participants' comments and answers are written down with paper and pencil. These notes, comments and answers are then further utilised in the preparation of more detailed questions in the interviews.

4.1.2.3 General Procedure

At the beginning of the test, all the participants are given written information about the evaluation process. This includes a brief definition of the systems (the conference management and payment systems) used in the test, and how these systems interact with each other. I also mention verbally that they are free to ask questions, or I will intervene if

[29] See Appendix 1 for the set of interview questions.

necessary. Afterwards, they are asked to fill out a short survey form about their backgrounds. This is followed by general instructions about this test. Within the framework of the think aloud technique, the participants are given a set of tasks to complete and are asked to talk through what they are experiencing when using the prototype. Their comments and my observations are used in the creation of more detailed questions for the interview, or are used as examples in the interview. After the tasks are completed, the participants are interviewed. Before beginning the interview session, I ask the participants to remember the story line (see Appendix 1) given at the beginning of the test process, and answer the interview questions as if they are in the situation related in the story.

The prerequisite information delivered to participants—the user background survey, task set, reports and interview questions—can be found in Appendix 1.

4.1.2.4 User Tasks

The user tasks are organized as a set of seven questions, and they are introduced with a story line that describes their role as conference manager and the situation they are in.

These tasks include the basic usage of the prototype: The participants are asked to find MONA's website, configure MONA and take out the reports. It is also asked whether the content of the reports is understandable, usable and relevant for their work.

The objective of these tasks is that when they are completed, the participants will be able to configure and navigate the prototype's capabilities perfectly and they will have all the necessary information about how MONA achieves its tasks on behalf of the user.

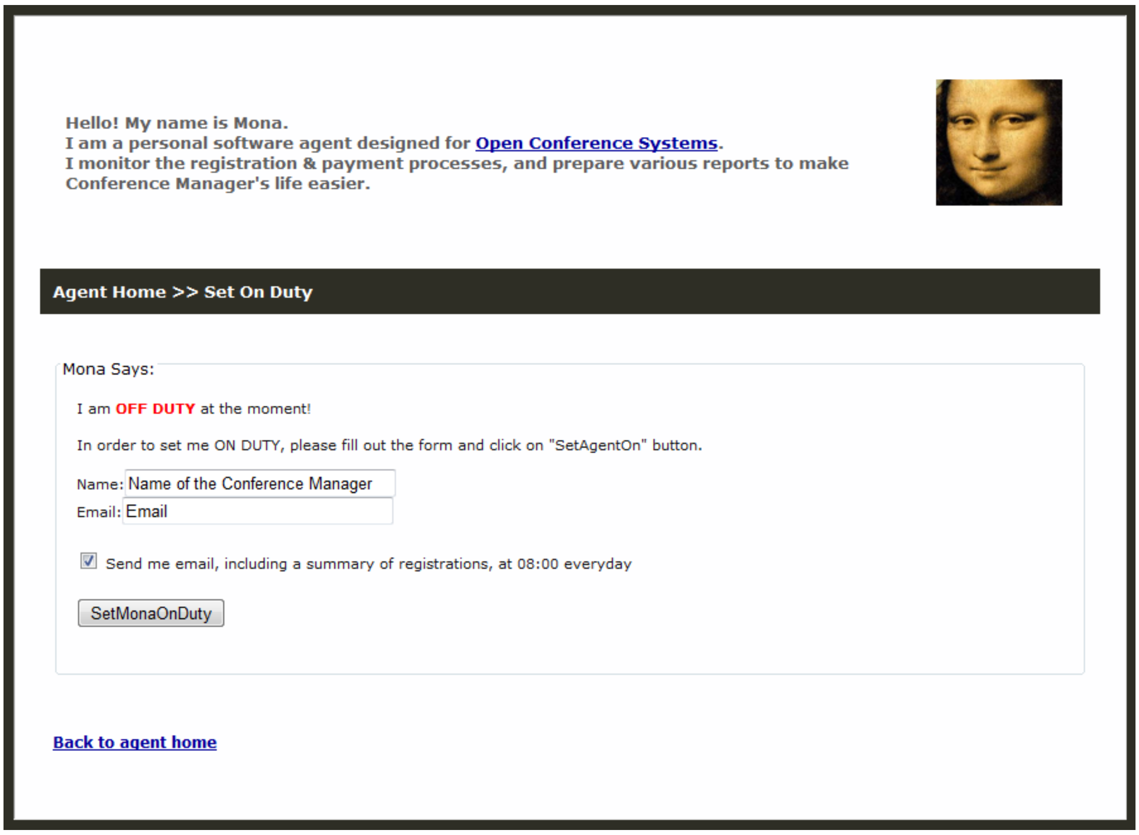
4.1.2.5 Configuration

Both the conference manager and the developer configure MONA.

Conference Manager:

The conference manager submits his or her name and email address, and decides whether he or she wants to get an email from MONA at the same time each day.

By clicking the "SetMonaOnDuty" button on the page, MONA is configured. This is shown in Figure 12.



Hello! My name is Mona.
I am a personal software agent designed for Open Conference Systems.
I monitor the registration & payment processes, and prepare various reports to make Conference Manager's life easier.

Agent Home >> Set On Duty

Mona Says:

I am OFF DUTY at the moment!

In order to set me ON DUTY, please fill out the form and click on "SetAgentOn" button.

Name:	Name of the Conference Manager
Email:	Email

[✓] Send me email, including a summary of registrations, at 08:00 everyday

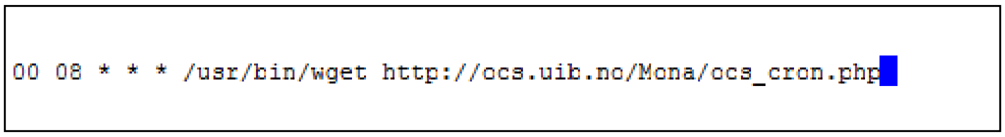
SetMonaOnDuty

Back to agent home

***Figure 12*: Configuration of MONA**

Developer:

From a technical standpoint, in order to let MONA send an email, the developer of MONA has to configure MONA by setting a crontab-job, which is used to let the server execute a particular PHP code file at the given time. Therefore, before each test, this configuration has been set to a convenient point in time so that the participants can receive the emails that are sent automatically by MONA. A sample crontab-job is shown in Figure 13.



```
00 08 * * * /usr/bin/wget http://ocs.uib.no/Mona/ocs_cron.php
```

***Figure 13*: Configuration of MONA's email service**

4.1.2.6 Interview Questions

Ten general questions are asked in the interview, which are also presented in Appendix 1. The questions are organized into four different categories to obtain data on usability and to help answer the research questions.

- ***Questions*[30] *1 to 7*** are designed to collect data in order to evaluate the usability of the prototype:
    - Q1: Overall, what do you think about using this prototype?
    - Q2: Do you find the prototype's basic functionalities valuable?
    - Q3: Is it easy for you to use the prototype and navigate between pages or assign different tasks to MONA?
    - Q4: How easily do you make inferences about how to use this interface (prototype), based on your previous experience?
    - Q5: Is the prerequisite information about the prototype (given at the beginning of this test and on the website) enough for you? What kind of prerequisite information do you need to use the product?
    - Q6: Which of the functions/tasks of MONA are easy to use and which will probably require either help or written documentation?
    - Q7: Do you think this prototype needs additional functions/ways of making it more user-friendly?
- ***Question 8*** is designed to answer research question 1:
    - Q8: Do you believe the fraud monitoring function of the prototype will be useful in your conference organization work? In which ways?
- ***Question 9*** is used to answer research question 2:
    - Q9: How much time do you think you will save by using MONA? (The registration list from the OCS and payment list are presented to the participants. These can be found in Appendix 1.)

[30] Questions 1–6 are adapted from Rubin and Chisnell (2008:30).

        - Q9a. If you had to check everything manually, without using MONA, how much time would you have to spend looking for possible fraud in the OCS?

In the question 9, the goal is to determine the effect of the prototype on the conference managers' work. After asking this question, I deliver the "registration report" from the OCS and the "transaction history" from the payment account to the participants. After they have read these reports, they are asked to estimate the time that would be needed if they had to do the same work manually. Time constraints necessitate this hypothetical question, as opposed to asking them to do the work manually. As a second step in this question, I present a printed copy of the "registration report" and the "fraud check report" taken from the prototype, before asking sub-question (9a), which is about detecting possible fraudulent activities in the system. The aim is to let participants evaluate the prototype in terms of both time use and efficiency. I expect that the participants comment on the changes regarding the quality of the completed work since time usage is an important parameter in their work.

- ***Question 10***, regarding the reliability of the prototype, is used to answer research question 3.
    - Q10: Do you think you can rely upon MONA to deliver the correct information? Do you have confidence in the information MONA provides? (The objectives of the test data and the results obtained by MONA are presented to the participants. These can be found in Appendix 1.)

Here, I equate the *reliability* of the prototype with the robustness of the program. The system should not be prone to "central points of failure" (Ehrler, Fleurke, Purvis, & Savarimuthu, 2006); the prototype should work properly and deliver the results of the tasks without failure.

With Question 10, the test data sheet is delivered to the participants. This sheet contains information about how the test data is constructed. After they have checked the test data sheet, I ask them to repeat task 5 (the fraud check) to check whether MONA's answers match the test data. This comparison lets participants evaluate the precision of MONA's results.

The interview questions overlap with each other. The aim is to keep the participants' work and situation awareness at a specific level, so that their understanding of the usability of this prototype increases over the course of the test.

4.2 Findings

In the following, the findings from the usability test will be presented in two main parts. First, I will focus on the usability of MONA, and then I will present the answers to the research questions.

4.2.1 Evaluation of the Developed Prototype's Usability

The participants were asked interview questions 1 to 7. Their answers indicate the following:

All the participants think that

- this prototype is useful, easy to learn and easy to use
- MONA's tasks are relevant for their jobs
- MONA works very fast and efficiently (in comparison to manual work)
- working with MONA will reduce the amount of time managers spend on the kinds of activities MONA performs automatically
- the preliminary information about MONA, given in the beginning of the test, is adequate for using this prototype
- MONA's tasks are relevant and adequate with respect to the scope of the prototype
- MONA should not take decisions, but is best used as a way of uncovering fraudulent activities

Comments:

- One of the participants wanted to know more about how MONA understands "fraud" situations, and one of the participants thinks that it would be better to have a written explanation of MONA's roles (as an adviser and observer). Two of the participants would like more functions which cover the whole conference process (e.g. online paper submission).
- Some of the reports come with links to searches on Google and Pipl. The participants unanimously agree that this is useful for their job.

- One of the participants noted that she likes to work with a "person" rather than a program, so she likes the name MONA.
- One participant thinks that a program to detect possible fraudulent activities will be very useful in her work. She gives an example from her own experience in managing a conference: In an international conference in Africa, the conference management system was attacked by a hacker to create many dummy records. She had to use a lot of time to clean the data and distinguish the real conference attendees from the fake ones. She thinks that if they had had MONA, it would have made this work much easier.
- One of the participants who has experience with OCS enthusiastically states that "this [is] the missing link!" MONA fills the gap between the payment system and the conference management system. She also thinks that MONA will be useful to her and her colleagues, such as the financial officer who deals with the payments.
- One of the participants thinks that without such tools, the conference cannot be fully under control.

Criticism:

The participants also offer comments on how to improve MONA:

- MONA should gain the ability to detect possible fraudulent activities not only in social sciences (academic) but also other kinds of conferences. One participant says that MONA should support international non-academic conferences (e.g. conferences to announce new products) as well as academic ones.
- All aspects of a conference should be embedded in MONA's activities. Two of the participants want MONA to act in all phases of a conference so that it will take on recurrent jobs.
- MONA should support not only conferences but also other kinds of international events that involve registrations. One of the participants wants to configure MONA in a way that it can support registrations at events such as summer schools/courses that accept international students.
- MONA should automatically connect to the account to get the account details. However, another participant claims that this is not a good idea, since the

account’s id and password combination is kept by the financial officer, and they do not want to put it in a program.

Observations:

All the participants are able to configure and use MONA without any intervention. They all complete the first part of the test quickly, within 5–13 minutes. They make comments more than they ask questions. It is interesting to see how some of the participants accept MONA as a “partner” rather than a program.

4.2.1.1 Discussion

The participants’ answers indicate that they consider MONA to be usable according to the following usability principles:

Learnability principle: The participants think that it is easy to learn how MONA works, set it on duty and assign it tasks.

Flexibility principle: The participants are satisfied with information that MONA carries from the systems (conference management and payment). They find the presented information understandable, helpful and usable.

Robustness principle: MONA does what it is built for and works efficiently.

It has been mentioned that MONA’s functional capabilities and functional flexibility could be enhanced so that it can support different kinds of registrations (e.g. school/university registrations) and different kinds of checks, such as when a person uses more than one email address. It has also been mentioned that all the processes of conference management should be checked by the agent one way or another—not merely fraud management—but since the agent does not become bored with doing the same thing over and over, and its work is not open to human error, some of the participants think that it would be better to use an agent for such tasks.

An important reflection is that most of the participants respond to the first interview question most comprehensively. With the other questions, they do not have much to add. As a result of this fact, it is better to refine the interview questions for future use.

4.2.2 Answering the Research Questions

4.2.2.1 Research Question 1: Can a monitoring agent tool be useful in supporting conference managers’ efforts to detect fraud in CMSs?

In order to answer this research question, the participants were asked interview question 8 (Do you believe the fraud monitoring function of the prototype will be useful in your conference organization work? In which ways?).

All the participants think that the monitoring agent, MONA, will be useful to their work since it produces reliable reports and does the work immediately so that they can act against fraudulent activities rather than losing time by doing the job manually. Three of the participants commented that MONA is reliable based on the test data, and two of them would like to test this prototype by themselves in order to give a final answer to this question.

The participants agree that there is a need for such tool to help detect possible fraud, and they also mention that they want a tool that does not take decisions itself, but merely supports their (conference managers’) work—just as MONA does.

One of the participants thinks that it is necessary to enhance MONA’s capabilities so that it can cover all aspects of a conference.

All in all, the participants accept MONA as a reliable and useful tool to support conference managers’ work in detecting possible fraud committed by conference registrants.

4.2.2.2 Research Question 2: What is the impact of agent technology integration on time and quality in medium-scale conference management systems? Does agent technology increase time effectiveness and enhance quality?

In order to answer this research question, the participants were asked interview question 9 (How much time do you think you will save by using MONA?) and question 9a (If you had to check everything manually, without using MONA, how much time would you have to spend looking for possible fraud in the OCS?).

All the participants think that MONA saves a lot of time compared with doing the same task manually. Most of the participants think that doing the task manually might take at least two hours and up to half a day. However, some of them claim that it might take much longer in an office environment, since interruptions are inevitable in the workplace. One of the participants even refuses to venture a guess because she is so often interrupted in her office.

One of the participants thinks that the list MONA produces is more reliable than the list that might be produced by a human being, adding that such a job, which demands a lot of time and attention, is vulnerable to human error. Another participant thinks that this task should not be done manually because the vital importance of finding fraudulent activities in conference registrations simply does not permit mistakes.

As a result, the participants think that working with a monitoring agent, MONA, saves time and energy in a work environment; it helps maximise time effectiveness since the participants use less time on the same work. As MONA presents the necessary information that can be gathered from the system, the managers can focus on the crucial parts of their work rather than spending their precious time collecting information. The participants also appreciate MONA's contribution to quality; doing the tasks manually opens the process to human error, but MONA is able to produce reliable outputs. In other words, by using MONA, the managers can improve in their work efficiency and ensure a higher degree of quality.

4.2.2.3 Research Question 3: Are the results of collaborating with the monitoring agent reliable? Are they convincing enough for further collaboration?

In order to answer this research question, the participants were asked interview question 10 (Do you think you can rely upon MONA to deliver the correct information? Do you have confidence in the information MONA provides?).

All of the participants find MONA reliable since they understand what it does (based on the test data) and they have experience/knowledge about the task.

The two participants who have the most experience in organizing conferences think that there is some chance for error (of MONA making a mistake), but they also want to spend more time on MONA to find out if they can trust its conclusions. As a point of departure, however, they find MONA reliable as well.

All in all, the participants have positive impressions of MONA's ability to detect fraudulent activities. They all say that they can <u>trust</u> MONA. Two of them have already asked whether they can use it in their upcoming conferences.

5 Conclusions and Future Work

Conclusions

Software systems are constructed according to, among other things, the business needs of a customer. However, it is not always the case that these systems are capable of fully meeting those needs. As Liskov and Guttag (2001:256) suggest, “More often customers do not fully understand what they want the program to do.” To this we might add that business needs can change and/or new business needs can emerge over time.

When the business is such that it is not well-represented in the current market for information systems, the business is more likely to be fragile and suffer as a result. Although it may not create problems on a daily basis, the available information systems carry the potential risk of failure or may be incapable of supporting the business.

The case of detecting fraudulent activities has a special importance because it is a critical task in any work environment. An example of this situation can be seen in conference management systems which do not have controls against fraudulent activities. Detecting such activities manually is time-consuming, and leads to an overload in the manager’s workflow.

In this project, a monitoring agent, MONA, is designed and developed to meet this need. The results of the evaluation show that this agent is useful in flagging possible cases of fraud so that the conference manager, who would otherwise have had to do the work manually, gains time and is able to take better decisions on such important cases. In addition, MONA is easy-to-use monitoring agent, which is an attractive feature to its potential users.

Although almost all of the participants are sceptic to MONA in the beginning of the evaluation, they have changed their attitude towards the end of the test. They are all interested in MONA and how she does the work. Besides, most of them are willing to use the agent in their work. They ask for new capabilities. One of the participants liked the name, MONA, and she accepted MONA as a part of her social-office life.

Moreover, this prototype is a good example of the thought that “some jobs should be done by the agents” (Chen, 2004). MONA never becomes bored of doing the same thing repeatedly, and it delivers reliable results according to its configuration and the data it reaches/uses in its environment.

Future Work

There are many design and development possibilities for MONA.

The logic of finding fraudulent activities can be enhanced with the feedback that is produced by possible and observed cases. The fraud indicators can be designed and formulated as a part of configuration that allows that the user might have a chance to choose the type of monitoring in the environment.

More automated skills can be added to the agent’s set of functionalities. For example, MONA can be integrated with the payment system so that it collects all the payment data itself, or it can be integrated with the webpages of e.g. UDI (the Norwegian Directorate of Immigration) so that it can obtain a list of the countries whose citizens need a visa to enter Norway. These integrations will lead to the use of up-to-date data in the process of detecting fraudulent activities.

Future work might also include an effort to decompose MONA into several specialised agents so that this prototype can be converted or rewritten into a multi-agent system (MAS) by redesigning the agents’ roles and tasks. These agents might include an interface agent (Wang, Wang, & Xu, 2005; Huhns & Singh, 1998), a resource agent (Huhns & Singh, 1998) and a monitoring agent to the model in which features such as learning capabilities may also included.

Another option can be to support MONA with mobile technologies so that it can be at the service of the coordinator and conference participants anywhere and at any time.

Moreover, users are often curious to understand how the agent arrives at its conclusions. Therefore, another possible improvement is to produce an explanation of the agent’s activities in a way that the agent will be accepted by its users at an increased level of trust.

Lastly, MONA’s flexibility can be enhanced, so that it can support not only conferences but also registrations in a broader sense.

References

Aart, C. v., & Tamma, V. (2008). *Agent mediated provision of insurance services. Two case studies: fraud and repairs*. Paper presented at the Proceedings of the 10th international conference on Electronic commerce.

Ambler, S. W. (2002-2009). Introduction to Test Driven Design (TDD). *Techniques for Successful Evolutionary/Agile Database Development.*

Bauer, B. (2002). UML Class Diagrams Revisited in the Context of Agent-Based Systems. *Agent-Oriented Software Engineering II* pp. 101-118.

Bernon, C., Cossentino, M., & Pavón, J. (2005a). Agent-oriented software engineering. *The Knowledge Engineering Review, 20*(2), pp. 99-116.

Bernon, C., Cossentino, M., & Pavón, J. (2005b). An Overview of Current Trends in European AOSE Research. *Informatica, 29*, pp. 379-390.

Bigus, J. P., & Bigus, J. (2001). *Constructing Intelligent Agents Using Java* (2nd ed.). New York, USA: Wiley Publisher.

Binder, W., Mori, J., Portabella, D., Tamma, V., & Wooldridge, M. (2005). State-of-the-art in Agent-based Services. *EU-IST Network of Excellence (NoE) IST-2004-507482KWEB, Deliverable D2.4.3* (KWEB/2004/D2.4.3/v1.0).

Bradshaw, J. M. (1997). Introduction. In J. M. Bradshaw (Ed.), *Software agents* (pp. 3-46): Menlo Park, Calif.: AAAI Press.

Bunch, L., Breedy, M., Bradshaw, J. M., Carvalho, M., Suri, N., Uszok, A., Hansen, J., Pechoucek, M., & Marik, V. (2004). *Software agents for process monitoring and notification*. Paper presented at the Proceedings of the 2004 ACM Symposium on Applied Computing.

Bussmann, S., Jennings, N. R., & Wooldridge, M. J. (2004). *Multiagent systems for manufacturing control: a design methodology*. Berlin: Springer.

Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Leal, F., Chainho, P., Kearney, P., Stark, J., Evans, R., & Massonet, P. (2002). Agent Oriented Analysis Using Message/UML. In M. J. Wooldridge, G. Weiß & P. Ciancarini (Eds.), *Lecture Notes in Computer Science: Agent-Oriented Software Engineering II* (Vol. 2222, pp. 119-135). Berlin: Springer.

Cao, J., Yang, J., Chan, W. T., & Xu, C. (2005). *Exception handling in distributed workflow systems using mobile agents.* Paper presented at the ICEBE 2005. IEEE International Conference on e-Business Engineering, 2005.

Chan, P. K., Fan, W., Prodromidis, A. L., & Stolfo, S. J. (1999). Distributed data mining in credit card fraud detection. *Intelligent Systems and their Applications, IEEE, 14*(6), pp. 67-74.

Chen, W. (2004). INFO282: Artificial Intelligence - Lecture notes.

Chen, W., Dolonen, J., & Wasson, B. (2003). Supporting Collaborative Knowledge Building with Intelligent Agents. *Knowledge-Based Intelligent Information and Engineering Systems* (pp. 238-244).

Chen, W., & Wasson, B. (2002). An Instructional Assistant Agent for Distributed Collaborative Learning. *Intelligent Tutoring Systems.*

Depke, R., Heckel, R., & Kuster, J. M. (2001). *Improving the agent-oriented modeling process by roles*. Paper presented at the Proceedings of the Fifth International Conference on Autonomous Agents.

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction* (3rd ed.). Essex, England: Pearson/Prentice Hall.

Ehrler, L., Fleurke, M., Purvis, M., & Savarimuthu, B. (2006). Agent-based workflow management systems (WfMSs). *Information Systems and E-Business Management, 4*(1), pp. 5-23.

FIPA: Foundation.for.Intelligent.Physical.Agents (2000). FIPA Modelling Area: Agent Class Diagrams, www.fipa.org, Geneva, Switzerland.

FIPA: Foundation.for.Intelligent.Physical.Agents (2003). FIPA Modelling Area: Environment, Version: 030412 15:00EST, www.fipa.org, Geneva, Switzerland.

Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language* (3rd ed. ed.). Boston, Mass. USA: Addison-Wesley.

Franklin, S., & Graesser, A. (1997). Is It an agent, or just a program? A taxonomy for autonomous agents. *Intelligent Agents III: Agent Theories, Architectures, and Languages* (pp. 31-35).

Giorgini, P., & Henderson-Sellers, B. (2005). Agent-Oriented Methodologies: An Introduction. In B. Henderson-Sellers & P. Giorgini (Eds.), *Agent-Oriented Methodologies* (pp. 1-18). London: Idea Group Publishing.

Grau, G., Cares, C., Franch, X., & Navarrete, F. J. (2006). *A Comparative Analysis of i*Agent-Oriented Modelling Techniques.* Paper presented at the Eighteenth International Conference on Software Engineering and Knowledge Engineering (SEKE'06).

Hayes, C. C. (1999 Jan-Feb). Agents in a Nutshell: A Very Brief Introduction. *IEEE Transactions on Knowledge and Data Engineering. 11*(1), pp.127-132.

Heo, J. S., & Lee, K. Y. (2005). *A multi-agent system-based intelligent control system for a power plant.* Paper presented at the Power Engineering Society General Meeting, 2005. IEEE.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly, 28*(1), pp. 75-105.

Hu, J., Zhang, C., Zhang, Y., Zhao, D., Chen, J., & Fang, C. (2008). *Research of Exception Handling in Workflow Management System Based on Agents Group.* Paper presented at the ICICIC '08. 3rd International Conference on Innovative Computing Information and Control, 2008.

Huhns, M. N., & Singh, M. P. (1998). All agents are not created equal. *IEEE Internet Computing, 2*(3), pp. 94-96.

IEEE.Std.1233 (1998 Edition). IEEE Guide for Developing System Requirements Specifications, NY, USA.

Imai, S., Yamada, A., Ueno, H., Nakamichi, K., & Chugo, A. (2006). Voice Quality Management for IP Networks Based on Automatic Change Detection of Monitoring Data. *Management of Convergence Networks and Services* (pp. 451-460).

Jami, S. I., & Shaikh, Z. A. (2007). *A workflow based academic management system using multi agent approach*. Paper presented at the Proceedings of the 11th WSEAS International Conference on Computers.

Jedermann, R., & Lang, W. (2006). *Mobile Java Code for Embedded Transport Monitoring Systems.* Paper presented at the Proceedings of the Embedded World Conference 2006, February 14-16, Nuremberg, Germany.

Jennings, N. R. (1999). Agent-Oriented Software Engineering. In F. J. Garijo & M. Boman (Eds.), *Multi-Agent System Engineering* (pp. 1-7).

Jennings, N. R. (2001). An Agent-based Approach for Building Complex Software Systems. *Commun. ACM 44*(4), pp. 35-41.

Jennings, N. R., Faratin, P., Norman, T. J., O'Brien, P., & Odgers, B. (2000). Autonomous agents for business process management. *Int. Journal of Applied Artificial Intelligence, 14*(2), pp. 145-189.

Jennings, N. R., & Wooldridge, M. (1998). Applications of Intelligent Agents. In N. R. Jennings & M. Wooldridge (Eds.), *Agent Technology: Foundations, Applications, and Markets* (pp. 3-28). London: Springler.

Kaminka, G. A., Pynadath, D. V., & Tambe, M. (2001). *Monitoring deployed agent teams*. Paper presented at the Proceedings of the Fifth International Conference on Autonomous agents.

Kim, K., Choi, I., & Park, C. (2010). A rule-based approach to proactive exception handling in business processes. *Expert Systems with Applications, 38*(1), pp. 394-409.

Kishore, R., Zhang, H., & Ramesh, R. (2006). Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems. *Decision Support Systems, 42*(1), pp. 48-78.

Lashkari, Y., Metral, M., & Maes, P. (Eds.). (1998). *Collaborative Interface Agents*. San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc.

Lauesen, S. (2005). *User interface design: a software engineering perspective*. Harlow: Pearson/Addison-Wesley.

Lee, J. (2000). Reactive-System Approaches to Agent Architectures *Intelligent Agents VI. Agent Theories Architectures, and Languages* (Vol. Volume 1757/2000, pp. 132-146): Springer Berlin/Heidelberg.

Liskov, B., & Guttag, J. (2001). *Program Development in Java: Abstraction, Specification, and Object-Oriented Design*: Addison-Wesley.

Liu, X., & Zhang, P. (2007). *An Agent Based Anti-Money Laundering System Architecture for Financial Supervision.* Paper presented at the WiCom 2007. International Conference on Wireless Communications, Networking and Mobile Computing, 2007.

Luck, M., McBurney, P., & Preist, C. (2003). *Agent Technology: Enabling Next Generation Computing: A Roadmap for Agent-Based Computing Version 1.0; AgentLink II*.

Mabry, S. L., Schneringer, T., Etters, T., & Edwards, N. (2003). *Intelligent agents for patient monitoring and diagnostics*. Paper presented at the Proceedings of the 2003 ACM Symposium on Applied Computing.

Maes, P. (1994). Agents that reduce work and information overload. *Commun. ACM, 37*(7), pp. 30-40.

Maes, P. (1997). Agents that Reduce Work and Information Overload. In J. M. Bradshaw (Ed.), *Software Agents* (pp. 145-164): Menlo Park, Calif.: AAAI Press.

Maheswaran, R. T., Tambe, M., Varakantham, P., & Myers, K. (2004). Adjustable Autonomy Challenges in Personal Assistant Agents: A Position Paper. *Agents and Computational Autonomy* (pp. 187-194).

McArthur, S. D. J., Booth, C. D., McDonald, J. R., & McFadyen, I. T. (2005). An agent-based anomaly detection architecture for condition monitoring. *IEEE Transactions on Power Systems, 20*(4), pp.1675-1682.

Menzies, T., Pearce, A., Heinze, C., & Goss, S. (2002). "What Is an Agent and Why Should I Care?" *Formal Approaches to Agent-Based Systems* (pp. 1-14).

Odell, J., Parunak, H. V. D., & Bauer, B. (2000). *Extending UML for Agents.* Paper presented at the Proc. of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence, AOIS Workshop at AAAI 2000 Austin, TX.

Padgham, L., & Winikoff, M. (2004). *Developing intelligent agent systems: A practical guide*. Chichester: Wiley.

Perry, W. E. (2006). *Effective Methods for Software Testing* (3rd. ed.). Indianapolis, IN, USA: Wiley Publishing, Inc.

Prodromidis, A. L., & Stolfo, S. J. (1999). *Agent-based distributed learning applied to fraud detection*. Paper presented at the Sixteenth National Conference on Artificial Intelligence.

Ragin, C. C. (1994). *Constructing social research: The unity and diversity of method*. Thousand Oaks, Calif.: Pine Forge Press.

Robertson, S., & Robertson, J. (2006). *Mastering the requirements process* (2nd ed.). Upper Saddle River, N.J.: Addison-Wesley.

Rosson, M. B., & Carroll, J. M. (2002). *Usability engineering: scenario-based development of human-computer interaction*. San Francisco, Calif.: Morgan Kaufmann Publishers.

Rubin, J., & Chisnell, D. (2008). *Handbook of Usability Testing* (2nd ed.). Indianapolis, IN, USA: Wiley Publishing, Inc.

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Englewood Cliffs, N.J.: Prentice Hall.

Shoham, Y. (1997). An Overview of Agent-Oriented Programming. In J. M. Bradshaw (Ed.), *Software Agents* (pp. 271-290): Menlo Park, Calif.: AAAI Press.

So, R., & Sonenberg, L. (2004). *Situation Awareness in Intelligent Agents: Foundations for a Theory of Proactive Agent Behaviour.* Paper presented at the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04).

Sommerville, I. I. (2004). *Software Engineering* (7th ed.). Boston: Pearson/Addison-Wesley.

Tveit, A. (2001). *A Survey of Agent-Oriented Software Engineering*. Paper presented at the NTNU Computer Science Graduate Student Conference.

Uckun, S. (1994). Intelligent system in patient monitoring and therapy management. *International Journal of Clinical Monitoring and Computing, 11*(4), pp. 241-253.

Urlings, P., Tweedale, J., Sioutis, C., & Ichalkaranje, N. (2003). Intelligent Agents and Situation Awareness. In V. Palade, R. J. Howlett & L. C. Jain (Eds.), *Knowledge-Based Intelligent Information and Engineering Systems* (pp. 723-733). Berlin Heidelberg: Springler-Verlag.

Vaswani, V. (2005). *How to Do Everything with PHP and MySQL*. Blacklick, Oh, USA: McGraw-Hill Companies.

Wang, M., & Wang, H. (2002). Intelligent Agent Supported Flexible Workflow Monitoring System. In A. Banks, P. J. Mylopoulos, C. C. Woo & M. T. Ozsu (Eds.), *Advanced Information Systems Engineering* (Vol. Volume 2348/2002, pp. 787-791): Springer Berlin Heidelberg.

Wang, M., & Wang, H. (2004). Agents and Web Services Supported Business Exception Management. In C. Zhang, H. W. Guesgen & W. K. Yeap (Eds.), *PRICAI 2004: Trends in Artificial Intelligence* (Vol. 3157, pp. 615-624): Springer Berlin/Heidelberg.

Wang, M., Wang, H., & Xu, D. (2005). The design of intelligent workflow monitoring with agent technology. *Knowledge-Based Systems, 18*(6), pp. 257-266.

Wang, Y., Xu, D., Wang, H., Ye, K., & Gao, S. (2007). *Agent-oriented ontology for monitoring and detecting money laundering process*. Paper presented at the Proceedings of the 2nd International Conference on Scalable Information Systems.

Wilhelm, W. K. (2004). The Fraud Management Lifecycle Theory: A Holistic Approach to Fraud Management. *Journal of Economic Crime Management 2*(2).

Wooldridge, M. (1997). Agent-based Software Engineering. *Software Engineering. IEE Proceedings, 144*(1), pp. 26-37.

Wooldridge, M. (1998). Agent-Based Computing. *Baltzer Journals*.

Wooldridge, M. (2002). Intelligent Agents: The Key Concepts *Multi-Agent Systems and Applications II* (pp. 151-190).

Wooldridge, M., & Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*.

Wooldridge, M., & Jennings, N. R. (1999a). The Cooperative Problem-Solving Process. *The Journal of Logic and Computation, Oxford University Press, 9*(4), pp. 563-592

Wooldridge, M., & Jennings, N. R. (1999b). Software Engineering with Agents: Pitfalls and Pratfalls. *IEEE Internet Computing, 3*(3), pp. 20-27.

Wooldridge, M. J. (2009). *An introduction to multiagent systems*. Chichester: Wiley.

Wooldridge, M. J., & Ciancarini, P. (2001). Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini & M. J. Wooldridge (Eds.), *Agent-Oriented Software Engineering* (pp. 1-28). Berlin Heidelberg: Springer-Verlag.

APPENDIX 1

A1 1 Evaluation forms

The forms that are used in the evaluation are listed below.

A1 1.1 General Information about the Evaluation Process

Open Conference Systems (OCS) is a free web publishing tool that will create a complete web presence for scholarly conferences. OCS allows you to:

- create a conference website
- compose and send a call for papers
- electronically accept paper and abstract submissions
- allow paper submitters to edit their work
- post conference proceedings and papers in a searchable format
- post, if you wish, the original data sets
- register participants
- integrate post-conference online discussions

In this master thesis, the focus is on registrations. In the registration process, it is also possible to collect fees via online payment (PayPal). In order to use PayPal, it is necessary to open a business account at this company.

MONA is an exception monitoring agent which is developed to detect (possible) fraudulent activities in the registration process. MONA's main goal is to collect necessary information in order to assist conference managers.

MONA checks both the information stored in OCS and the payments. Payment information is stored in a file, which includes all the transactions of the PayPal account. Moreover, MONA can perform different tasks such as sending emails and performing searches on the names of registrants via google.com and pipl.com.

A1 1.2 User Background Survey[31]

Thank you for agreeing to participate in this study. Before we begin, it will be useful for me to know more about your background—your experience with conference management tools and internet technologies. This will help me better understand your interactions with and reactions to the system. Remember that all personal data will be treated confidentially, and the results will be reported with no identifying information.

Name: ____

Occupation: ____

Age: ____

Gender: ☐ F ☐ M

Education: ☐ BA ☐ MA ☐ PhD Other: ______________________________

Field: ☐ Natural Sci. ☐ Social Sci. ☐ Humanities Other: __________________

Have you used a conference management system...

as a conference attendee?	☐ YES	☐ NO	
scope of the conference(s):	☐ Local	☐ National	☐ International
as a conference manager?	☐ YES	☐ NO	
scope of the conference(s):	☐ Local	☐ National	☐ International
Have you ever used PayPal?	☐ YES	☐ NO	

Is there anything else I should know about your interests or background? Please describe briefly:____

____

____

[31] This part is adapted from Rosson & Carroll (2002:259).

A1 1.3 General Instructions for Usability Testing for Monitoring Agent Prototype[32]

- In the next 30 minutes or so, you will be carrying out seven tasks.
- The tasks will be introduced with a story line that describes your role and the situation you are in.
- Note that I have intentionally omitted some details about the task steps so that I can determine how well the system guides your interactions with it.
- If you are confused at any point, please just make your best guess about how to proceed, using the information that you have been given. I will intervene if necessary to help you make progress.
- At the start of each task, please say out loud, "Beginning task," followed by the number of the task. When you are done, please say, "Task complete."
- Also, please remember to **think out loud** as you work. It is very important for me to understand your goals, expectations, and reactions as you work through the tasks.
- Any questions?

[32] This part is adapted from Rosson & Carroll (2002:259).

A1 1.4 Specific Task Instructions for Monitoring Agent Prototype[33]

Background to tasks 1 through 7

- Imagine that you are a conference manager of an international conference being held in Bergen.
- The conference management system is OCS, and it is expected that there will be more than 100 attendees at this conference.
- You are both very busy and have an important meeting about this conference tomorrow where you will be presenting an overview of the attendees and payments. At tomorrow's meeting, you know that you will have to report any fraudulent activities that might have occurred in the registration process.
- You know that you can use MONA to monitor OCS.

Task 1:

Go to MONA's website, www.ocs.uib.no/Mona/, and read about what it can perform for you.

Task 2:

Set MONA On Duty by registering your name and email address. Assign MONA the task of sending you an email about every day at ________.

Task 3:

Get MONA to bring up a summary of all registrations.

Task 4:

Get MONA to send you an email with a summary of all registrations. Check to see if you have received this email.

Is the content of the email understandable and useful?

[33] This part is adapted from Rosson & Carroll (2002:259).

Task 5:

Get MONA to bring up a list of registrations based on payment transactions. This report will include fraud attempts if there are any.

Task 6:

Get MONA to bring up the list of actions it has performed.

Task 7:

Check to see if you have received the email, which is scheduled to be sent at ________.

Is the content of the email understandable and useful?

A1 1.5 Data Collection Form for Monitoring Agent Prototype[34]

Date: ______________________________

Participant Id: ______________________________

Task No: ______________________________

Start time: ______________________________

End time: ______________________________

Comments made by the participant

____

____

____

____

Errors or problems observed (including assistance offered)

____

____

____

____

Other relevant observations

____

____

____

____

____

____

____

____

____

[34] This part is adapted from Rosson & Carroll (2002:259).

A1 1.6 Test Data

Objectives:

It is assumed that every conference attendee pays for him- or herself. Based on this assumption, if the payer is different than the attendee, this might indicate fraudulent activity. This is reflected in the stored data in OCS and payments in three different ways:

CASE 1: The payer's email address is different than the attendee's email address.

Data: "Cathrine Ling" from China uses a different email address than the one she uses in OCS. It is possible that her fee has been paid by someone else. *The email address has a key importance in the payment process.*

Expected error message: Fraud?

CASE 2: The payer's full name is different than the attendee's full name.

Data: "John Smith" from Nepal uses "Sanjeev Parar" as the full name to pay the conference fee, although he uses the same email address.

Expected error message: Check!

CASE 3: The payer's full name and email address are different that the attendee's full name and email address.

Data: "Ama Dayo" from Sudan uses "Astray Novakantu" as the full name and a different email address to pay the conference fee.

Expected error message: Fraud?

Summary of Registrations:

Total number of registered attendees: 42 (Paid: 36, Not paid: 6)

Total number of fraud possibilities: 2

Total number of registrations that have to be checked: 1

A1 2 Interview Questions[35]

Questions 1–6 are adapted from Rubin and Chisnell (2008:30).

1. Overall, what do you think about using this prototype?
2. Do you find the prototype's basic functionalities valuable?
3. Is it easy for you to use the prototype and navigate between pages or assign different tasks to MONA?
4. How easily do you make inferences about how to use this interface (prototype), based on your previous experience?
5. Is the prerequisite information about the prototype (given at the beginning of this test and on the website) enough for you? What kind of prerequisite information do you need to use the product?
6. Which of the functions/tasks of MONA are easy to use and which will probably require either help or written documentation?
7. Do you think this prototype needs additional functions/ways of making it more user-friendly?
8. Do you believe the fraud monitoring function of the prototype will be useful in your conference organization work? In which ways? (research question 1)
9. How much time do you think you will save by using MONA? (research question 2)

The registration list from the OCS and the payment list are presented to the participants.

   a. If you had to check everything manually, without using MONA, how much time would you have to spend looking for possible fraud in the OCS?

10. Do you think you can rely upon MONA to deliver the correct information? Do you have confidence in the information MONA provides? (research question 3)

The objectives of the test data and the results obtained by MONA are presented to the participants.

[35] Questions 1–6 are adapted from Rubin and Chisnell (2008:30).

APPENDIX 2

A2 1 User Manual for MONA

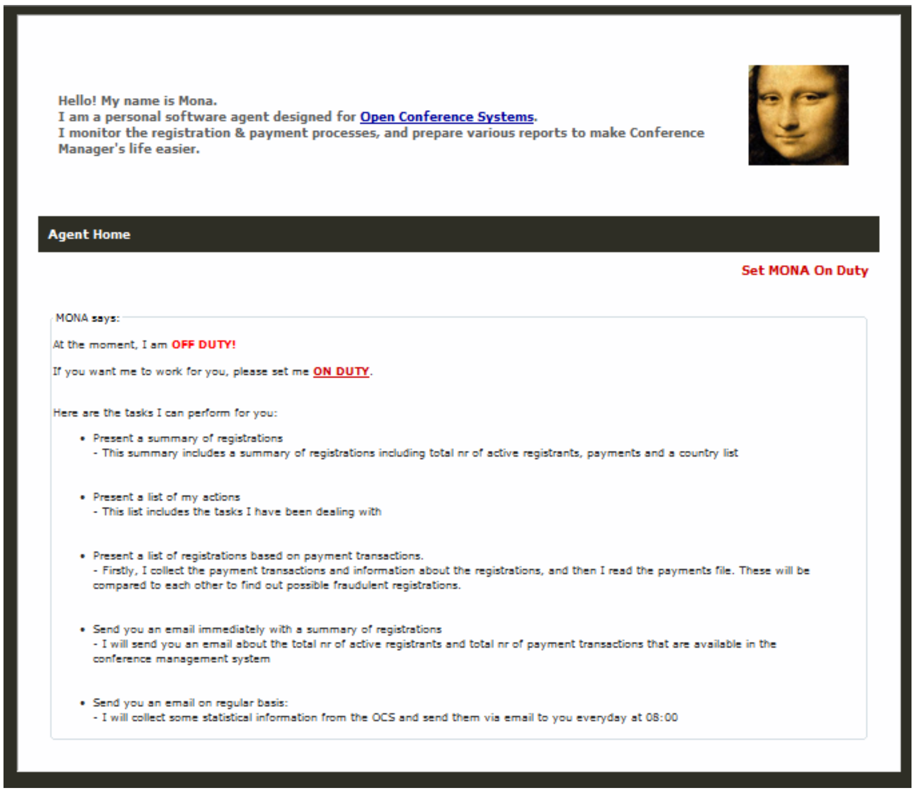
MONA, an exception monitoring agent, has been designed and developed as a part of master's thesis at the University of Bergen. It works as a personal assistant and helps to detect possible fraudulent registrations at conferences. For more information about MONA, visit www.ocs.uib.no/Mona.In order to use this agent, it is necessary to use OCS (Open Conference Systems) as the main conference management tool. For more information about OCS, visit the PKP web site at http://pkp.sfu.ca/?q=ocs.

A2 1.1 Installation

Download the installation package from http://sourceforge.net/projects/monitoringagent/ and extract the MONA archive to the desired location in your web documents directory. Review `config.php` and `readme.txt` for additional configuration settings.

A2 1.2 Log into MONA

Mona is now live at your website. In order to reach it, use www.yourwebsite.com/Mona.This first page of the application is shown below:



Hello! My name is Mona.
I am a personal software agent designed for Open Conference Systems.
I monitor the registration & payment processes, and prepare various reports to make Conference Manager's life easier.

MONA says:

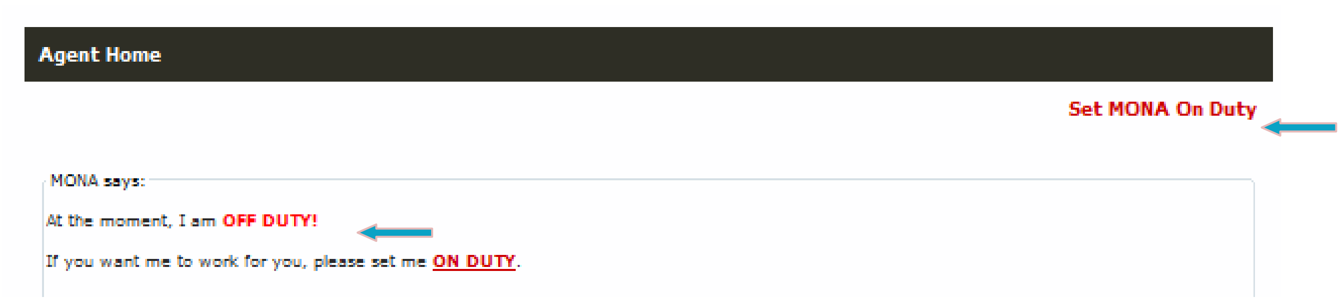
At the moment, I am **OFF DUTY!**

If you want me to work for you, please set me **ON DUTY**.

Here are the tasks I can perform for you:

- Present a summary of registrations
  - This summary includes a summary of registrations including total nr of active registrants, payments and a country list
- Present a list of my actions
  - This list includes the tasks I have been dealing with
- Present a list of registrations based on payment transactions.
  - Firstly, I collect the payment transactions and information about the registrations, and then I read the payments file. These will be compared to each other to find out possible fraudulent registrations.
- Send you an email immediately with a summary of registrations
  - I will send you an email about the total nr of active registrants and total nr of payment transactions that are available in the conference management system
- Send you an email on regular basis:
  - I will collect some statistical information from the OCS and send them via email to you everyday at 08:00

A2 1.3 Configuration


Agent Home

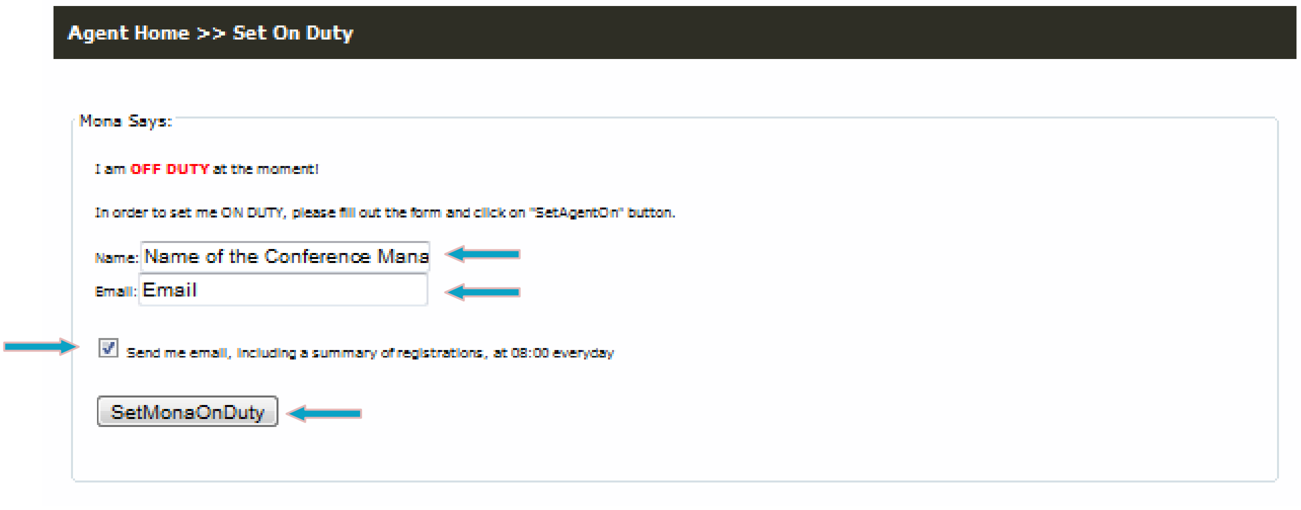
Set MONA On Duty

MONA says:

At the moment, I am OFF DUTY!

If you want me to work for you, please set me ON DUTY.

By clicking **ON DUTY** or **Set MONA On Duty**, the configuration page will be shown.


Agent Home >> Set On Duty

Mona Says:

I am OFF DUTY at the moment!

In order to set me ON DUTY, please fill out the form and click on "SetAgentOn" button.

Name: Name of the Conference Mana

Email: Email

☑ Send me email, including a summary of registrations, at 08:00 everyday

SetMonaOnDuty

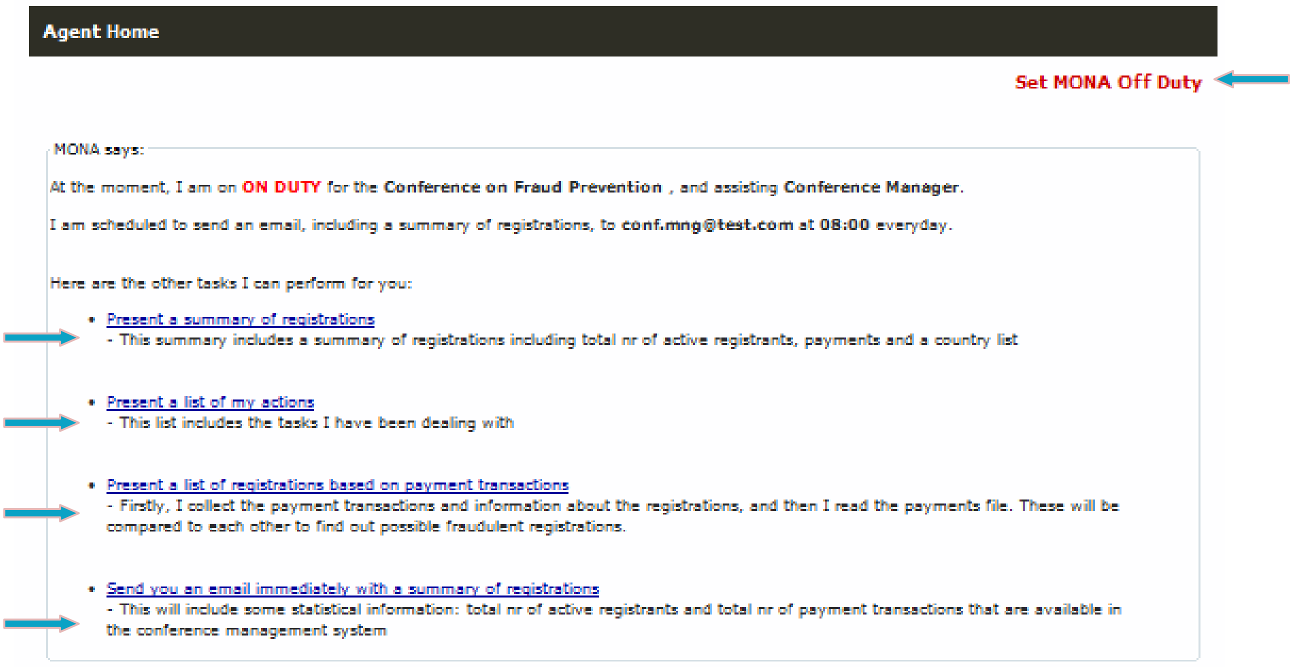
Fill out the **name** and **email** fields.

If the user wants to get an email from MONA every day, the **Send me email** option should be ticked. This email includes the summary of registrations as of 08:00 AM.

When the **SetMonaOnDuty** button is clicked, MONA is at the user's service and can be assigned to different tasks.

IMPORTANT NOTE: All the names that are used in registrations are either the product of the author's imagination or, if real, used fictitiously without any intent to describe their actual conduct. This has one exception, and it is my adviser: Weiqin Chen

A2 1.4 Assigning Tasks


Agent Home

Set MONA Off Duty

MONA says:

At the moment, I am on **ON DUTY** for the **Conference on Fraud Prevention** , and assisting **Conference Manager**.

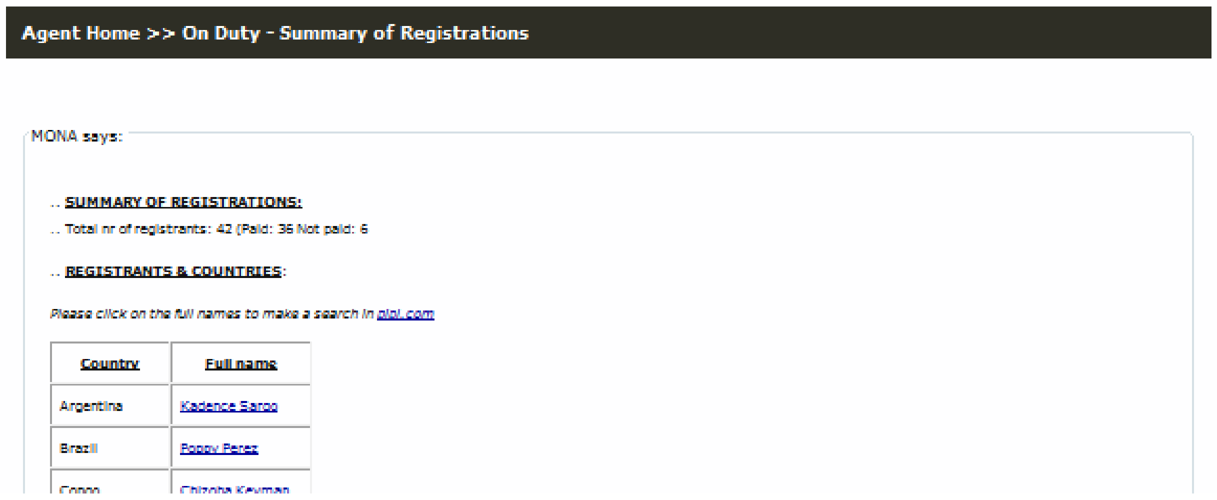
I am scheduled to send an email, including a summary of registrations, to **conf.mng@test.com** at **08:00** everyday.

Here are the other tasks I can perform for you:

- Present a summary of registrations
  - This summary includes a summary of registrations including total nr of active registrants, payments and a country list
- Present a list of my actions
  - This list includes the tasks I have been dealing with
- Present a list of registrations based on payment transactions
  - Firstly, I collect the payment transactions and information about the registrations, and then I read the payments file. These will be compared to each other to find out possible fraudulent registrations.
- Send you an email immediately with a summary of registrations
  - This will include some statistical information: total nr of active registrants and total nr of payment transactions that are available in the conference management system

In order to assign MONA to different tasks, click on the links on this page.

A2 1.4.1 Summary of registrations


Agent Home >> On Duty - Summary of Registrations

MONA says:

.. **SUMMARY OF REGISTRATIONS:**

.. Total nr of registrants: 42 (Paid: 36 Not paid: 6

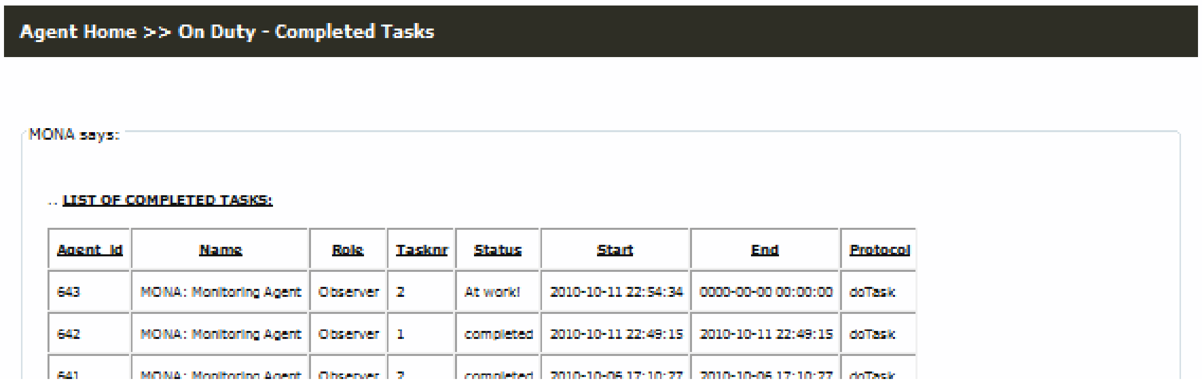
.. **REGISTRANTS & COUNTRIES**:

Please click on the full names to make a search in pipl.com

Country	Full name
Argentina	Kadence Sargo
Brazil	Poppy Perez
Congo	Chizoba Keyman

With this task, MONA presents a report containing all the registrations in the conference management system. The report includes the country and full name of the attendees. It is possible to search the names of the attendees on pipl.com by clicking the name links in this report.

A2 1.4.2 List of MONA's actions



Agent Home >> On Duty - Completed Tasks

MONA says:

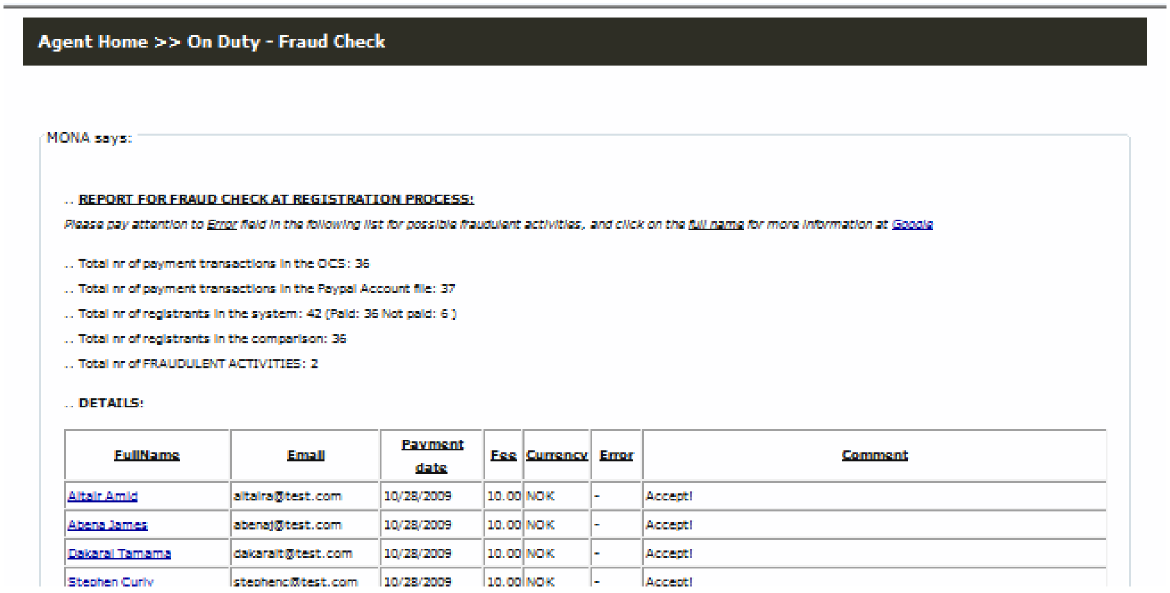
.. **LIST OF COMPLETED TASKS:**

Agent_Id	Name	Role	Tasknr	Status	Start	End	Protocol
643	MONA: Monitoring Agent	Observer	2	At work!	2010-10-11 22:54:34	0000-00-00 00:00:00	doTask
642	MONA: Monitoring Agent	Observer	1	completed	2010-10-11 22:49:15	2010-10-11 22:49:15	doTask
641	MONA: Monitoring Agent	Observer	2	completed	2010-10-06 17:10:27	2010-10-06 17:10:27	doTask

With this report, MONA shows the task history with details of what it has done, when it started each task, how long it has taken, whether it has been successfully completed or not, the protocol name it has used, its role is in each task and which agent has been assigned to each task.

A2 1.4.3 List of registrations based on payment transactions

NB! In order to get a result in this report, the payment file should be uploaded to the **upload** folder on the server.



Agent Home >> On Duty - Fraud Check

MONA says:

.. **REPORT FOR FRAUD CHECK AT REGISTRATION PROCESS:**

Please pay attention to Error field in the following list for possible fraudulent activities, and click on the full name for more information at Google

.. Total nr of payment transactions in the OCS: 36

.. Total nr of payment transactions in the Paypal Account file: 37

.. Total nr of registrants in the system: 42 (Paid: 36 Not paid: 6)

.. Total nr of registrants in the comparison: 36

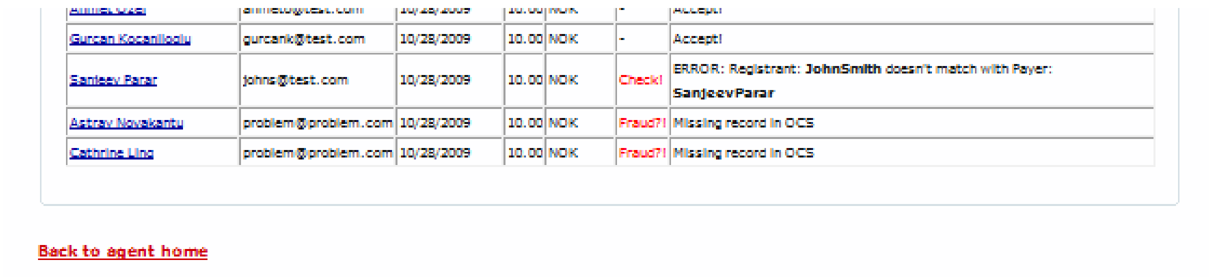
.. Total nr of FRAUDULENT ACTIVITIES: 2

.. **DETAILS:**

FullName	Email	Payment date	Fee	Currency	Error	Comment
Altair Amid	altaira@test.com	10/28/2009	10.00	NOK	-	Accept!
Abena James	abenaj@test.com	10/28/2009	10.00	NOK	-	Accept!
Dakarai Tamama	dakarait@test.com	10/28/2009	10.00	NOK	-	Accept!
Stephen Curly	stephenc@test.com	10/28/2009	10.00	NOK	-	Accept!

Continues on the next page

Contiuned

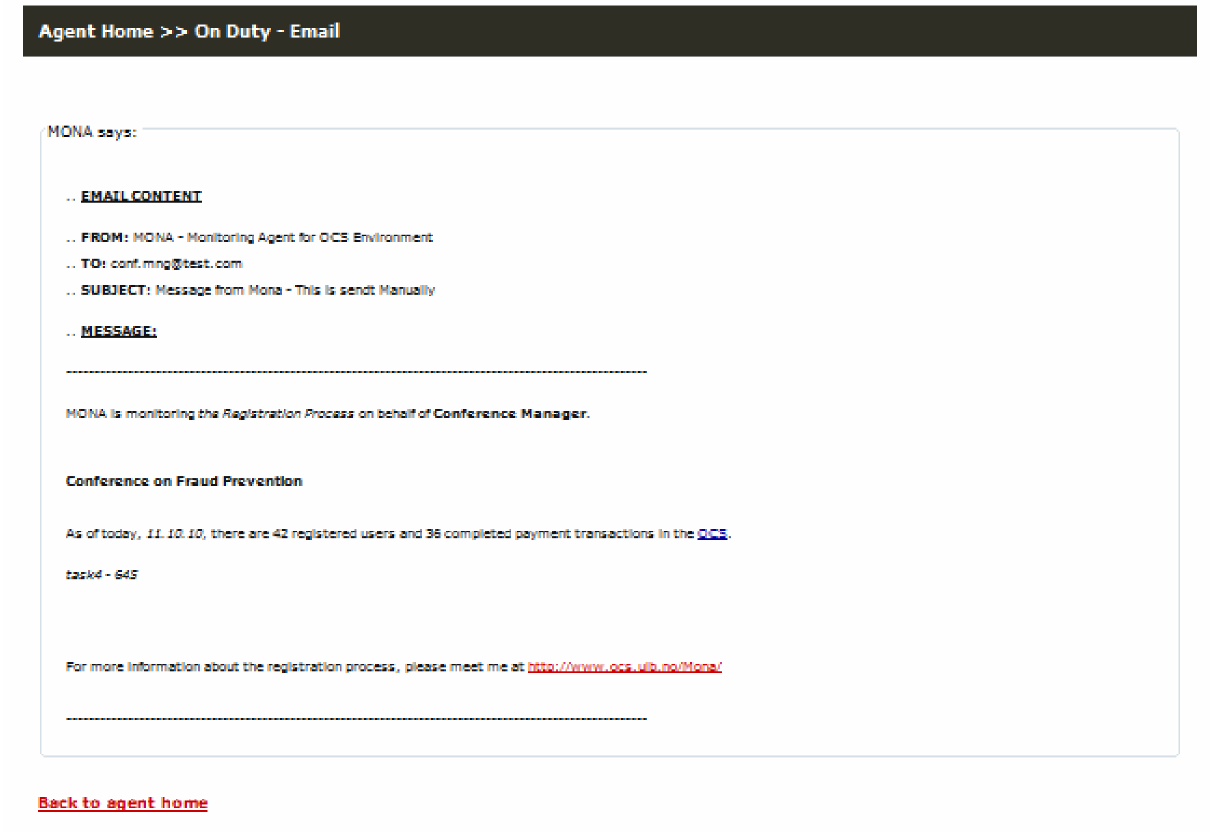


Ahmet Ozel	ahmeto@test.com	10/28/2009	10.00	NOK	-	Accept!
Gurcan Kocanlioglu	gurcank@test.com	10/28/2009	10.00	NOK	-	Accept!
Sanjeev Parar	johns@test.com	10/28/2009	10.00	NOK	Check!	ERROR: Registrant: **JohnSmith** doesn't match with Payer: **SanjeevParar**
Astrav Novakantu	problem@problem.com	10/28/2009	10.00	NOK	Fraud?!	Missing record in OCS
Cathrine Ling	problem@problem.com	10/28/2009	10.00	NOK	Fraud?!	Missing record in OCS

Back to agent home

MONA presents a detailed report about the registration process. The report begins with an overview of registrations, which is followed by details concerning whether the person should be accepted as an attendee or checked before acceptance. The records that should be checked are written in red font and they are marked with an explanatory comment about how the check can be done. It is also possible to launch a Google search of the person by clicking the name link.

A2 1.4.4 Email a summary of registrations



Agent Home >> On Duty - Email

MONA says:

.. **EMAIL CONTENT**

.. **FROM:** MONA - Monitoring Agent for OCS Environment

.. **TO:** conf.mng@test.com

.. **SUBJECT:** Message from Mona - This is sendt Manually

.. **MESSAGE:**

MONA is monitoring *the Registration Process* on behalf of **Conference Manager**.

Conference on Fraud Prevention

As of today, *11.10.10*, there are 42 registered users and 36 completed payment transactions in the OCS.

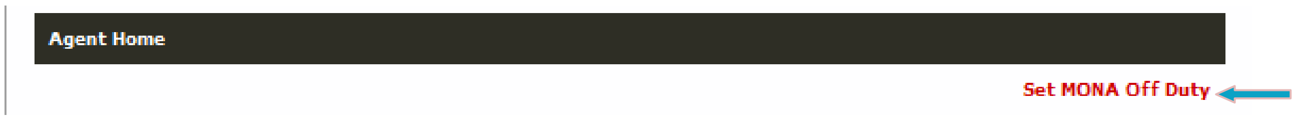
task4 - 645

For more information about the registration process, please meet me at http://www.ocs.uib.no/Mona/

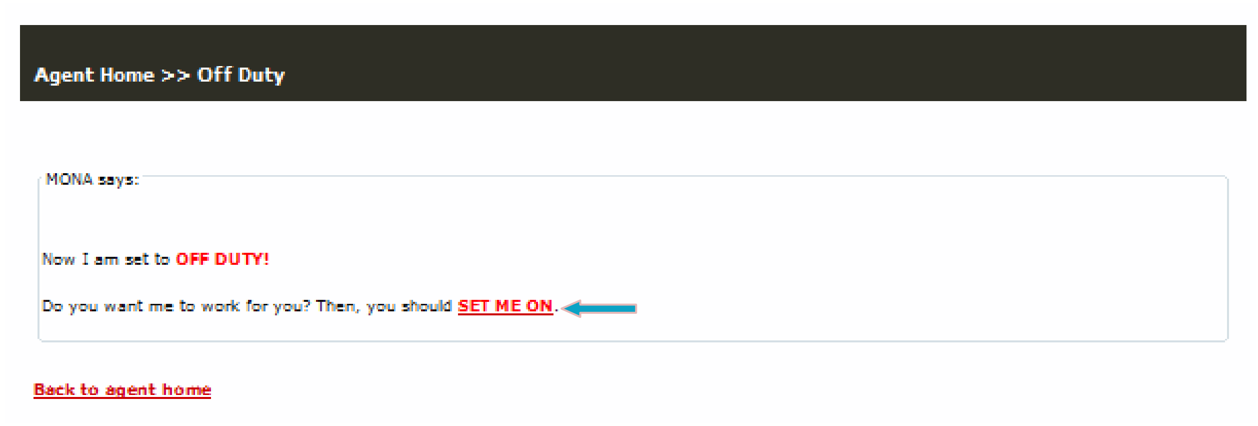
Back to agent home

With this task, MONA presents a summary of registrations and the content is sent to the user by email.

A2 1.4.5 Set MONA Off Duty



By clicking the **Set MONA Off Duty** button on the agent's home page, the user can stop MONA from accepting tasks.


Agent Home >> Off Duty

MONA says:

Now I am set to OFF DUTY!

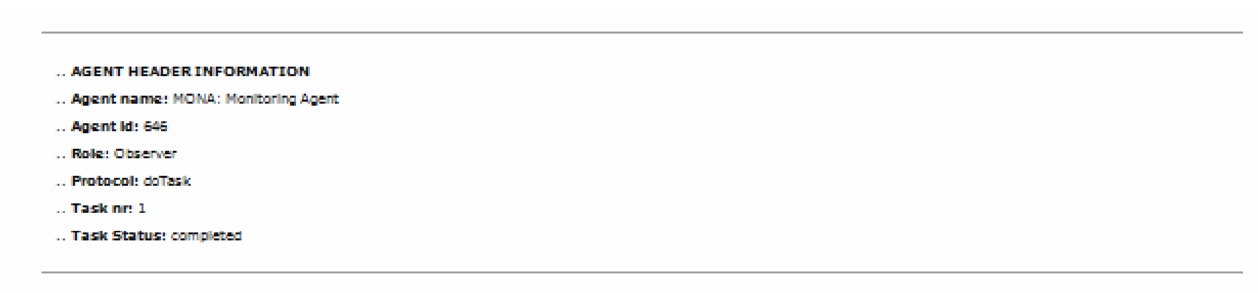
Do you want me to work for you? Then, you should SET ME ON.

Back to agent home

If user wants to work with MONA, it can be reactivated with the **SET ME ON** button.

A2 1.4.6 Agent Header Information

Every report includes the agent header information at the end of the report. This shows information about the state of MONA at the actual time of delivery.


```
.. AGENT HEADER INFORMATION
.. Agent name: MONA: Monitoring Agent
.. Agent Id: 646
.. Role: Observer
.. Protocol: doTask
.. Task nr: 1
.. Task Status: completed
```