Faculty of Science, Technology and Communication

Proactive Dynamic Community of Practice

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information and Computer Sciences

Author:
Remus-Alexandru Dobrican

Supervisor:
Prof. Denis Zampunieris

Reviewer:
Ass. Prof. Steffen Rothkugel

Advisor:
Prof. Pascal Bouvry

August 2012

Acknowledgements

Firstly, I would like to thank Professor Denis Zampunieris for proposing this Master Thesis, for welcoming me into his team, for his constant help during this internship and for giving me an opportunity to continue with my research in fields like Proactivity and Communities of Practice.

I am extremely grateful to Sandro Reis, Denis Shirnin and Sergio Marque Dias for their help, assistance and guidance. Their expertise and advice has been extremely valuable, and without them this project would not have been possible.

Special thanks are due to Ms. Josiane Geisler and to Ms. Octavie Modert, Luxembourg's Minister of Culture, for agreeing to extend my scholarship for the first year of my Master program.

Thanks are extended to Professor Steffen Rothkugel and to Professor Pierre Kelsen, the director of my Master program, for helping me at the beginning of my Master with recommendation letters needed for my scholarship extension.

I also wish to thank my friends and colleagues at the University of Luxembourg: Masoud Tabatabaei, Diana Marosin, Cristina Ghet, Sergio Sousa, Avikarsha Mandal, Vaishnavi Rajendran, Azemina Husovic and Walter Bronzi.

Finally, my special thanks go to my parents, Alexandru and Elena Dobrican, which supported me morally, emotionally and financially from the beginning of my Master program.

Abstract

The main purpose of this Master Thesis was to invent and design Proactive Scenarios that could help students from the University of Luxembourg to improve their social experience on the Moodle™ e-learning platform through the automatic perception of who the user is, where does he/her come from, what are his/her online activities or what do users have in common, and through constant help, guidance and assistance generated by the system with the help of Proactivity.

The first part of this work was dedicated to reading, analyzing and comparing state-of-the-art papers, articles and tools in research fields like Proactivity, Communities of Practice, Situation/Context Awareness and Learning Management Systems, followed by the creation and validation of the Proactive Cycle meant to group students, which are inscribed in the same study program or coming from the same city, into social communities where the practice can take multiple shapes.

Contents

List of Figures

List of Tables

CoP Community of Practice

OCoP Online Community of Practice

VCoP Virtual Community of Practice

LMS Learning Management System

PLMS Proactive Learning Management System

GUI Graphical User Interface

Moodle Modular Object-Oriented Dynamic Learning Environment

VLE Virtual Learning Environment

LIFO Last In First Out

RRS Rules-Running System

MTA Meta-Scenario

PS Proactive System

PAM Proactive Awareness Management

SA Situation Awareness

TSA Team Situation Awareness

ISA Individual Situation Awareness

SM Situation Management

Chapter 1

Introduction

A short resume of what was done in this thesis is provided below. It contains a brief overview of all the chapters of this Master thesis as well as the different ways they interconnect. At the beginning of each chapter, a small introduction is given to summarize the ideas of that chapter.

We show, by reviewing the research done until now, that common software systems lack proactivity and ways of grouping people for obtaining better results. We worked on defining the characteristics of an ideal proactive system which is ready to implement Communities of Practice. Then, we propose a solution for transforming a very popular Learning Management System (i.e. Moodle) into a proactive system where students are organized in social groups according to their city of origin and their study formation. And finally, we explained the reasons for choosing a Learning Management System, especially why Moodle in particular.

Chapter 2 - Background Information is dedicated for giving the necessary background information on research fields like *Proactivity*, *Learning Management Systems*, *Communities of Practice* and *Situation/Context-Awareness*. It highlights key terms, explains important concepts and try to answer to the most frequently asked questions concerning these fields.

Chapter 3 - Review of the State of the Art, contains a personal review of the related work relevant to this Master Thesis. Only the most pertinent research, related to this study's investigations, is presented across the four sections: **Proactivity**, **Proactive Learning Management Systems**, **Communities of Practice** and **Situation and Context Awareness**. Issues are pointed out in order to have a clear perspective of what is to be addressed and later on, solved.

Chapter 4 - Theoretical Background presents the most important part of this thesis. Three sections are dedicated for describing the ideal proactive software platform for deploying Communities of Practice (CoPs), the solution for transforming a software system into the ideal system proposed before and the proper environment for implementing this solution.

Chapter 5 - Practical Implementation is concentrated on the practical implementation part proposed in section 2 - *Solutions to have an ideal system* of chapter 4 - Theoretical Background. It is shown shortly what elements need to be added on Moodle's client-side and server-side, and on the proactive engine's side.

And finally, in chapter 6 - Conclusions the most significant findings are synthesized, together with the summary of contributions and the future work. Future research present directions to be followed like applying proactive CoPs, finding new patterns to apply proactive rules and creating a Graphical User Interface (GUI) for the proactive engine, in order to continue the ideas generated in this investigation.

Chapter 2

Background Information

A very powerful influence over the reader's comprehension of this material, especially concerning the research subjects and their key elements, is given by the background information acquired at the beginning of this thesis. Prior knowledge is known to be extremely important for the reading process as it helps the reader to understand the major points of several topics, what is known about them and how are these topics connected.

This introductory chapter is specially written for the reader which did not meet yet with concepts such as *proactivity*, *learning management systems*, *communities of practice* or *situation-awareness*. It provides explanations of the different concepts and ideas used in all the other chapters. These explanations are not very elaborated but contain only the essence of each topic.

Why is the background knowledge so important?

According to on-line dictionary [TheFreeDictionary, 2012], the main role of background knowledge is highlighted with a simple, but concise definition - "information that is essential to understanding a situation or problem". To continue this idea, it helps the person who reads to focus on terms, ideas, examples and definitions associated with a certain research fields.

It also gives a broad overview of the topics, introduces key issues and makes the reader more familiar with areas where only a few general ideas are known. It offers the possibility to choose the angle for approaching the various concepts and research topics, in particular because lots of details are explained in chapter 3 - Review of the State of the Art and it is difficult to fit in all the details without having an overview of the big picture.

Answering important questions related to a certain subject is considered one of the best ways of introducing a study or explaining relevant matters as people learn at varied rates, with multiple styles and by different methods.

2.1 Proactivity

Most of the current research focuses on defining the various types of proactive behavior. Researchers have troubles when study the different aspects of proactivity such as its mechanisms, situation antecedents, manifestations and its consequences. Many inconsistencies and inefficiency problems occur when approaches are proposed for constructing, evaluating and improving a system's proactive behavior [Grant and Ashford, 2008].

What is proactivity?

The exact definition of proactivity could be defined as "self-initiated and future-oriented action that aims to change and improve the situation or oneself" [Crant, 2000]. Seen from another point of view, proactivity is a solution for the majority of systems, which, regrettably, are quite static and user-centered, depending a lot on decisions coming from a human being. Unfortunately, these computational systems are very reactive, which means they are designed to wait for a human's implicit command and to act according to it. Instead, with the help of proactivity, they can take their own decisions, in the interest of each user, as they become more sensitive to the user's intentions and goals.

Is proactivity a new research field or a new concept?

Proactivity is considered to be nearly a new field and it is closely related to other research fields like *Artificial Intelligence*, *Situation-Awareness* and *Ubiquitous Computing*. They share a lot of common features and sometimes they complete each other when working together for achieving the same goals.

More about how the concept of proactivity was created and how it evolved across the coming years can be found in section 1 - Proactivity and section 2 - Proactive Learning Management Systems (PLMS) of chapter 3 - Review of the State of the Art.

Is the behavior of a system proactive or not?

Proactive behavior refers to systems that make things happen. Being proactive depends on a lot of things: first and the most important property is to be aware of particular situations that may appear (i.e. situation-awareness), to anticipate issues or unwanted events, to guide and help its users to reach their aims, and finally to be able to take decisions without waiting for a direct intervention of a human or without being instructed before.

In [Grant, 2007] Grant proposes three main characteristics for defining a proactive behavior:

- **Anticipation** – Does it anticipate any kind of situation, rather than just statically react to it?
- **Self-initiative** – The system does not wait to be asked for making an action, neither require detailed instructions.
- **Change-oriented** – Because the term proactive means to be in charge of the situation, this feature refers to the fact that a system should cause things to happen, than just react to them.

2.2 Learning Management Systems

Even though many papers proposed multiple ways to ameliorate the current web-based educational system, the e-learning software remained static because they still needed user actions, such as a "click", to trigger additional actions and intelligent modules.

While many corporations are facing issues like lacks of budget and resources, they are still looking for solutions to have consistent training possibilities and learning procedures for their members.

According to [KSERVER™, 2012], a good e-Learning strategy is based on the following three key features:

- A relevant and intriguing learning content;
- An adequate and efficient environment for delivering knowledge;
- The capacity of managing the learning of its members;

What is a LMS?

A very important term present all across this thesis is *LMS*. It is intensively used in this work because it represents the appropriate platform where concepts such as *Proactivity*, *Communities of Practice* and *Situation-Awareness* can be applied. A LMS is an excellent tool for managing, evaluating and tracking results, while giving communities of people the opportunity to have a centralized content available via a personalized GUI with proper technological support.

LMSs, also known as e-learning platforms, are dedicated software environments which provide virtual educational services. It was among the products which emerged as a result of having assisted e-learning instructional programs. Teachers have the possibility of organizing their virtual courses and creating diverse activities for learners, while handling administrative tasks.

To summarize it, a LMS is considered these days one of the best tools for a company or educational institutions, like schools and universities, to plan, implement and evaluate the learning process of their people.

What are the benefits of using a LMS?

LMS have evolved in such a way that they are capable of doing almost everything a teacher would need. Not only are they excellent for distance learning in schools, colleges and universities, but they can be used with no problems for other purposes like corporate training, testing employees before they get hired or for breaking a huge amount of administrative work into smaller pieces.

LMSs are valuable to organizations, institutions and companies that are looking for a way of regularly managing the learning process and the development of their members, increase competency, reduce the rate of losing key people and finally, improving their overall output and productivity. They provide a compact and organized learning environment, reporting and tracking tools for increased performance, instant evaluation methods and advanced possibilities of managing the users. Only the fact that an e-learning software would save many of the costs otherwise necessary for training people is to be considered an argument on its own.

The administrator is offered great control for arranging the virtual classroom as learning management systems are highly customizable. Even students have the option of customizing their learning environment as they want to. The systems are taking care of the organization process, from inscribing new users to setting deadlines and reminders for certain tasks.

Taking into consideration all the major advantages of a LMS, it can be seen that their behavior is still missing some important features that are identified in proactive behaviors. Members of an e-Learning community would be grateful for more personalized help, immediate support and proper assistance from the LMS, based on the smart analysis of the user's activity and intentions.

2.3 Communities of Practice

One of the main purposes of this master thesis is to analyze the use of proactivity in the context of communities of practice. That is why a small introduction is required for the key concepts and ideas of communities of practice before analyzing the research that has been done already in this field.

The term *Communities of Practice* exists from a long time ago, but is regarded as a quite new and interesting concept. Two of the key aspects of this vast research field describe how to handle knowledge and, how to manage and improve the learning process. More and more people, as well as big organizations, are considering using CoPs to obtain better financial and social results by improving their output. One famous example is the company Xerox which, after certain estimations, saved US100 million by creating its Eureka database [Brown and Duguid, 2000].

One of the best ways of explaining a particular concept is to answer a couple of important questions related to that subject. On his site [Wenger, 2006] , Etienne Wenger offers a brief introduction for communities of practice by addressing four questions. The brief introduction contains one of the most significant explanations for communities of practice that was published until now. Following the answers provided in Etienne Wenger's description, a small summary, containing only the main ideas, is presented for the above questions.

What are communities of practice?

To answer the first question, the following definition is given: "Communities of practice are groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly" [Wenger, 2006]. The important part comes with the reason which brings these groups of people together – they either want to learn, work or exchange information between them. Three main structural characteristics are proposed to identify communities and to distinguish them from neighborhoods. These are: *the domain*, *the community* and *the practice*.

The *domain* represents the area of knowledge which brings people together, creates a sense for their actions, while inspiring and guiding them through the whole process. *The community* is actually the groups of people that interact and collaborate together for improving their learning skills. And finally, *the practice* is the central point around which the community evolves, by sharing and developing the knowledge gathered from different activities/actions.

What do communities of practice look like?

An essential step is to identify CoPs and to distinguish them from other forms of communities. Some CoPs are really small, others are quite big, usually having a nucleus formed by the most important members. They differ as they can be located all over the globe or just locally, some have online support, while others require their members to be physically present. They can be widely known or just hidden.

The point is that CoPs existed since people started learning together. An individual currently belongs to a number of communities – at work, at home, in his business. He will still be part of many CoPs during his lifetime.

Where does the concept come from?

The concept of CoP was proposed by Etienne Wenger and Jean Lave while researching apprenticeship as a learning model. They realized that communities are crucial for the learning process and they can be seen almost everywhere. People were already applying this concept in a natural way without even knowing that they are applying a pattern for getting together and handling their knowledge.

Where is the concept being applied?

The answer to the last question contains the key reason for which communities of practice are so important in today's society. This concept can be applied in education, in the business sector, for governments and civic life at both professional and non-professional levels. A big number of organizations across the world realized that knowledge is a critical resource and that social groups are essential in their structure. In fact there is almost no big corporation worldwide which did not use some kind of form of communities.

Governments and agencies have adopted CoPs from merely the same reasons. They often need to address problems in different sectors like security, education and health. This requires lots of efforts in sharing knowledge and acquiring new information.

In education, institutions like schools and universities have adopted new ways of improving the learning process. This is by far away the most important field where CoPs can be applied because it can add a lot of value to the whole process of acquiring new knowledge, behaviors, skills and values.

As for the business sector and for associations, members are struggling to improve their level of cooperation, to focus on very particular goals and to bring new value to their work by interacting with other members and sharing available resources.

And finally, on the web, CoPs are influenced by the emerging technologies. The fast evolution in this domain has made it easier to create and manage communities. Various learning systems are created all over the Internet in many scales and for different societies.

For future reading, a complete pdf file, with many more details about the answers to the above questions, is available on Etienne Wenger's site [Wenger, 2006].

2.4 Situation and Context Awareness

Scientists and programmers from the entire globe have been trying to make computers more adaptable to user needs. This implies that computers would sense, perceive and react to different external factors and situations. The situation/context awareness process underlines the importance of being familiar with the surrounding environment.

When a software system has this property it can efficiently provide knowledge about all the resources being used, information about running tasks or processes and, it can detect, locate and describe potential threats or issues. Even though detecting problems is very useful, the other capabilities are far more important for this study.

The next-generation software needs situation awareness to make more intelligent decisions and to stay in control. Applications that have an increased level of awareness are able to change their behavior according to each situation.

Definition of Context

Three classes of context are mentioned in [M. DEBES and SEITZ, 2012]: user context, terminal context and communication network context. A combination of these classes is encountered in the majority of applications and services to better serve their purposes.

What Is Situation or Context Awareness?

A raw definition could be that Situation and Context Awareness are resumed to the ensemble of tips, techniques and strategies for evaluating and responding to different situations or to special events.

Simply described by Eris S. Toner in [Eric S. Toner, 2012], situation awareness depends on understanding what happens around, both internally and externally. The process of understanding is more complex as it not only relies on gathering information but collecting precise information. Only the necessary amount has to be cumulated considering it has to be analyzed, and based on this analysis the right decisions must follow. To summarize it, situation or context awareness is composed of two stages: the understanding phase and the decision phase.

What is Context-Aware Computing?

This form of computing refers more to moving people and mobile devices as they can examine the computing surroundings and react to changes of the environment. According to [Schilit et al., 1994a] there are three key aspects of a context: the location, the systems involved and the available resources. Ubiquitous computing is strongly connected to this way of computing.

Chapter 3

Review of the State of the Art

The starting point in proactive computing is considered to be the article of David Tennenhouse, written in 2000 [Tennenhouse, 2000]. Opposite to interactive computing, which represented the vision of J.C.R Licklider's in [Licklider, 1960], proactive computing was thought to be omnipresent and human-supervised. Following the same idea of proactivity, Antti Salovaara and Antti Oulasvirta started describing proactive systems as working on behalf of the user and following their own initiative in [Salovaara and Oulasvirta, 2004].

In 2006, Denis Zampunieris published a new series of articles, in which he was describing some theoretical aspects of proactivity [Zampunieris, 2006a] together with their practical implementation [Zampunieris, 2008] in an LMS. These two papers are very important because they contain the basis for our proactive system. A more detailed description of the rules engine, the algorithms behind it and examples of rules are offered in chapter 3.2 – "Proactive LMS".

Later on, in 2007, a more complete visual software tool and architecture were described in [Alami et al., 2008] for offering a better on-line educational environment. These tools were then used to collect results and to analyze statistics based on these results in papers [Coronado and Zampunieris, 2008] and [Coronado and Zampunieris, 2010]. Professor Zampunieris and his team processed a series of three studies at the University of Luxembourg, in a real learning environment.

Recent research in the PLMS field, part of Denis Shirnin's PhD thesis [Shirnin, 2012] is trying to show a strong connection between proactivity and the fact of succeeding in a learning management system.

Communities of practice existed since a long time but it was only in 1998 that Wenger and Laye gave them an official name, a form and multiple structures depending on their goal [Wenger, 1998]. The literature review's aim [Wenger, 2006], [Wenger, 2001] and [Wenger, 2001] focuses more on a particular form of CoP, called online communities of practice (OCoPs) or virtual communities of practice (VCoPs). Their benefits are described in [Wenger et al., 2002] and [Probst and Borzillo, 2008]. Situation Awareness (SA) is a very important characteristic of any advanced software platform. SA's purpose change from analyzing the context change of individuals, in 1994, [Schilit et al., 1994b] to examining the major factors of human-machine interaction in [Shu and Furuta, 2005] , proposing models that understand and predict certain situations [Jakobson et al., 2007] and implementing awareness features in software systems [Oscar Nierstrasz and Rothlisberger, 2008].

3.1 Proactivity

In 2000, David Tennenhouse thought it was time for a change in the computer science research community and that is why he proposed a new way of computing called *"proactive computing"* [Tennenhouse, 2000]. The article came as an answer to the increasing number of networked interactive computers, which outnumbered humans a long time ago, and to the sinks of information.

Because the rate of growth of the embedded devices and the volume of data were increasing rapidly, the environment needed to change as well – from human-mediated sources of information to computer gathered information. This goes a bit in the direction of ubiquitous computing, a concept defined by Mark Weiser in his article in 1991 [Weiser, 1991]. Meanwhile ubiquitous computing focuses on using and connecting all the smart devices of our daily life, proactive computing can serve for analyzing the data collected by the smart sensors.

Tennenhouse proposed three new research directions for proactive computing: "Getting physical", "Getting real" and "Getting out".

In the chapter "Getting physical", key aspects are pointed out on how a broad proactive infrastructure should be deployed worldwide in order to monitor and shape the physical environment. Smart sensors and actuators are supposed to be part of all the proactive nodes. Networks supporting the data exchange of the nodes should be very cheap and flexible, easily integrating newly developed architecture in the near future. Specific software will allow users to manipulate data and to perform actions like multi-tasking and statistical analyses in a faster-than-real-time manner. Instead of using sensors for specific tasks and processes, users will be able to use specific dynamic sensors for network-based services.

The second chapter, "Getting real", takes care of the computational part, meaning the time it takes for all the operations to be performed, normally at very high speeds. This is important because the time the systems will produce an output from the data collected from the input sensors will be faster than humans are used to understand and react. The main idea is to propose some standards that have to be met in design and analysis for proactive computers because the environment being sensed will be also changed.

In the last chapter – "Getting out" – the author places the users on top of the loop, at a level where they have to supervise and maintain the proactive systems. Another point is that individual nodes should be accessible to users in case some particular information is needed. This raised the following issues: how will humans will be able to interact with the proactive systems via an interface when the operations are performed very fast, how humans should be less and less involved in programming and in software design and finally how to reuse and connect remaining software.

When computers are capable and authorized to make decisions in our behalf, questions about safety, accountability and delegation are still very important and remain to be answered.

In contrast with the article of David Tennenhouse, where the author focuses mainly on how should a proactive environment look from the technical point of view, Antti Salovaara and Antti Oulasvirta analyzed in their article [Salovaara and Oulasvirta, 2004] proactive systems from the user's perspective.

From their point of view, when speaking about the concept of proactivity, a system is supposed to have the following two main characteristics:

i To work for the user or pro-user;

ii To be able to take decisions without the user's explicit allowance;

A couple of key attributes were proposed in order to distinguish proactive computing systems, such as: real-time operation, world model, hypothesized goal state, sensitivity to future alternatives and taking initiative. These attributes are important because the system needs to track, monitor and analyze real-time ongoing activity in different contexts, meanwhile being opened to the user's input and taking decisions based on his/her choices.

This article treats proactivity in the context of resource management. More concrete, the need of people for resources, in order to achieve their daily goals, is the main area of interest. Based on this idea, the system can advise the user in different modes such as: preparing, optimizing, advising on the use of resources, manipulating, inhibiting and finalizing resources. The order of the modes is not that important and can be easily rearranged depending when the intervention of the proactive system takes place.

The computer takes an active part in handling all the modes. Either it initiates new resources, modifies resources to make them more suitable for the user's needs, advises and emphasizes the best solutions, changes the attributes of the resources, prevents the use of some assets or finally finishing the process of using extensively some resources.

When the article was written, little research was done in the field of proactivity, so the research was very helpful. This was considered as a strong basis for future proactive scenarios that could assist the users in finding a better solution for their needs.

Both of the papers [Tennenhouse, 2000] and [Salovaara and Oulasvirta, 2004] presented above try to propose concepts and to define specific terminology for proactivity in order to help researchers advance more in this field.

3.2 Proactive Learning Management Systems (PLMS)

In the paper "Implementation of a Proactive Learning Management System", published in 2006, a new and different type of learning management system was proposed by Denis Zampunieris [Zampunieris, 2006a]. The purpose of the system was to guide and assist users for better on-line interactions in educational and training environments. This was done by analyzing the user's interactions in a regular, continuous and automated manner, as well as the actions generated by the LMS itself.

Figure 3.1 contains the pseudo-code of the initial dynamic rules-based system.

The structure of a dynamic rule, proposed by Zampunieris, was composed of five main parts: **data acquisition**, **activation guards**, **conditions**, **actions** and **rules generation**. An abstract class called *"AbstractRule"* was defined in order to serve as a model for all the other rules. For gaining simplicity and efficiency, the rules were not parameterized, except for the rule generation procedure. Despite all of these constraints, useful and efficient rules could be further developed.

In the data acquisition phase, which was the first phase to be performed, a rule procured information from the LMS, which was then used in the other phases (e.g. getting from the database the current time). The second part, also called the activation guard, was used to determine if the next two phases will be triggered or not. The third part, "conditions", was similar to the second one and was made to check if the fourth phase, "actions", would be performed. The "actions" would contain a sequence of specific instructions which would be executed if the "conditions" would allow it. And finally, the fifth one, "rules generation", was always processed and it had the purpose to allow other rules to be generated. The architecture of the rules system allowed the creation of rules that will run over a long period of time.

A Last In First Out (LIFO) list was used by the proactive LMS in order to store the

```
i. repeat for each data acquisition request DA
   a.perform DA
   b.if error then raise exception on system manager console and go to step vii
     else create new local variable and initialize it with result of DA
ii.create new local Boolean variable "activated" initialized to false
iii.repeat for each activation guard test AG
   a.evaluate AG
   b.if result == false then go to step vi
     else if AG == last activation guard test
          then activated = true
iv.repeat for each conditions test C
   a.evaluate C
   b.if result == false then go to step vi
v.repeat for each action instruction A
   a.perform A
   b.if error then raise exception on system manager console and go to step vii
vi.repeat for each rule generation R
   a.perform R
   b.insert newly generated rule as the last rule of the system
vii.delete all local variables
viii.discard rule from the system
```

Figure 3.1: The initial rules engine in pseudo-code

rules and two important parameters, which modified the state of the rules engine, were defined at runtime: **"F"** that measured the time frequency of the activation periods and **"N"** that measured the number of rules per activation period. When the rules engine started, a check was done to see if the state of the system was activated or not. After the activation, the LIFO rules list was being executed by the rules engine. It ran one rule at a time and it was based on each rule's rank. A rule was either removed from the system if it was already used or it was reactivated by cloning itself.

A set of four rules was described in the paper as examples of different proactive employments and applications. The result was that the Proactive Learning Management System (PLMS) helped and guided e-learners, provided notifications, reacted to the actions or to the lack of actions of the users and was able to manage itself automatically, using in the same time a dynamic set of rules.

In the second paper published in 2006 [Zampunieris, 2006b], Zampunieris identified a couple of efficiency issues regarding the dynamic rules-based system proposed in [Zampunieris, 2006a]. He proposed a new evaluation method for the rules-running system, called lazy evaluation, which tried to prevent the use of time-costly requests to LMS's database. The lazy evaluation technique is used to "delay the evaluation of an expression until the value of this is actually required (non-strict evaluation) and which also avoids repeated evaluations (sharing)" [Wikipedia, 2012a].

More concretely, in this paper, an example of automatic management of the LMS, triggered by the proactive system, was given. This example was the second rule case, where a rule gets the number of connected users each five minutes and then stores it in a certain database table. In fact, the problem was that the request *"nb_ users = sys.getNumberOfConnectedUsers()"* was done in the data acquisition phase and it was only needed in the actions phase, just before the following request is made *"sys.dbStore (table = 'statistics', values = time nb_ users++)"*.

The major change from the technical point of view from the first paper [Zampunieris, 2006a] is that the structure of the variables was changed and consisted of three fields,

e.g. **"<name, definition, value* >"** instead of only two *"<name, value>"*. This was important because when a rule was run, in the data acquisition phase, the variables were created but no values were assigned to them.

Afterwards, in the other phases, when the value of a variable was needed, there were two possibilities. It was either used directly, if the value* field was different than the value to_be_computed or the data was used and then stored as the value of the variable, after evaluating the expression of the variable, i.e. the *definition field*. Even though the efficiency issue looked small for the above example, it becomes significant when the system has to handle big sets of rules. Taking in account that the average number of queries to the LMS database is equal to two or three requests per rule, the total number of database request will affect the system's performance. Finally, the purpose of the article was to decrease the average response time of the Learning Management System to the user's actions.

A more complete description of the architecture of the LMS was given in [Zampunieris, 2008] together with a set of new software tools which provided a virtual support and a training on-line environment for the proactive e-Learning Management System. In figure 3.2, which appeared in [Alami et al., 2008], the authors gave a general overview of a LMS system that works with the proactive RRS. A typical LMS would come without the RRS, which works in parallel with the LMS's server. They both communicate with the database in order to retrieve and store information such as the current state of the system and so on.

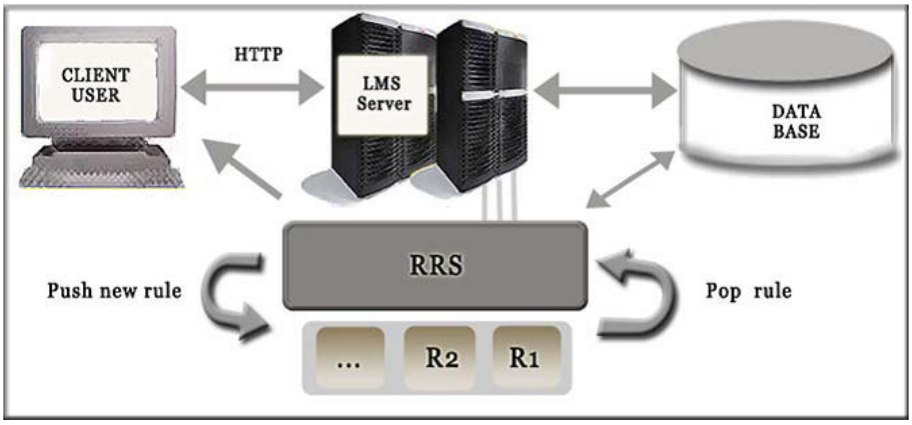

Figure 3.2: Implementation of the proactive part in the LMS by the RRS

A visual interface, composed of the "Message Zone" shown in figure 3.3 and the "Message Manager" shown in figure 3.4, was developed and integrated in the proactive system in order to show how it would interact with the user. The implementation was done is such a way that would help, guide and inform the user in different matters, without interfering in he's actions. The "Message Zone", a Flash application designed to show messages, hints and warning, was displayed in the header of the web system and remained always visible during the user's session.

The "Message Manager" was shown when the user clicked on one of the items in the message list. This meant that a more detailed description of the item was seen on the screen and the user was able either to save or discard it.

The first rules running-system architecture was developed with the help of sockets, the C programming language and Flash. The C language was chosen because of its advantages in terms of speed and simplicity, the sockets because an opened connection needed to be maintained such that both sides could become servers and the Flash because

Figure 3.3: The Message Zone

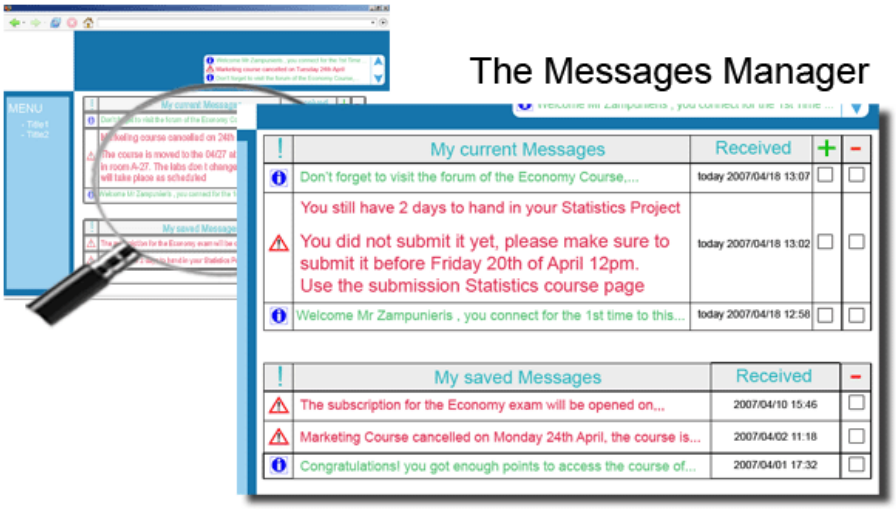

Figure 3.4: The Message Manager

it was widespread, cross platform and could manage socket connections. In figure 3.3, a more complete overview of the primary RRS Architecture was presented.

The authors also proposed a sequence diagram for a better understanding of the system. In figure 3.6 a full execution cycle was shown, which started with a message from the rule located on the server side and ended on the client's screen. Afterwards, the actions or the lack of actions from the user were registered on the server side.

The first RRS architecture proposed by Zampunieris, Alami and Casel had possible efficiency problems when evaluating a large set of rules. The issue was better described together with the solution for solving it in [Zampunieris, 2008]. They are also discussed above, as a short resume of paper [Alami et al., 2008].

Two new papers were published by professor Zampunieris and professor Coronado in 2008 [Coronado and Zampunieris, 2008] and in 2010 [Coronado and Zampunieris, 2010]. These papers analyzed the results of the practical implementation of the proactive system proposed in [Zampunieris, 2006b] and described above. They were also showing that students with proactive behavior had better chances to achieve better results in a LMS environment. This was firstly shown in a research conducted by Kickul in 2006 [Kickul and Kickul, 2006].

The first paper, [Coronado and Zampunieris, 2008], presented the results of a case study developed at the University of Luxembourg in 2006-2007 at the level of a Bachelor class, with more than 40 students. Statistics were used to find out how proactivity could influence the interaction of the students inside the LMS.

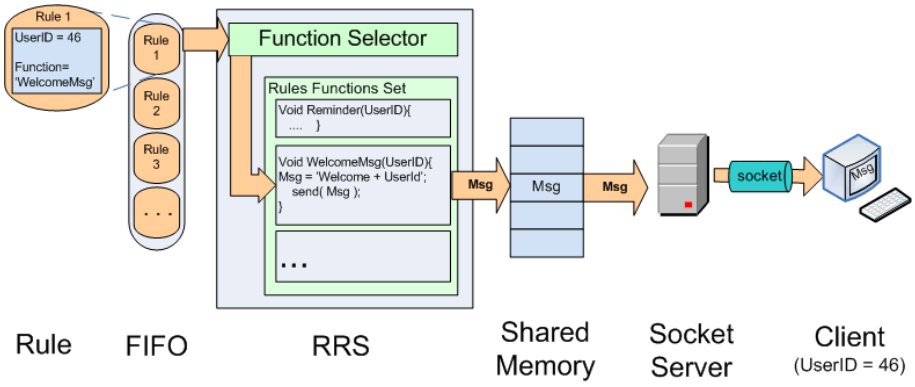

Figure 3.5: The structure of the initial RRS

The concept of "early proactive triggering" was introduced as well as different methodological approaches which study this new idea. Because the learning process requires a continuous step-by-steep engagement from the students, scenarios were created to monitor the degree of involvement of students inside the LMS. By tracking the student's activities and comparing it to proactive scenarios the LMS could warn users about different things (i.e. an email was sent to those students which did not take enough part in activities inside the LMS).

Three main parts were required for detecting early activity: proactive scenarios, proactive categories and proactive rules. The idea was that while scenarios described special situations, i.e. when a student decreases his/her learning process, proactive categories were used for grouping multiple scenarios and proactive rules for the interaction between the LMS and the users.

Results showed a high percentage for the correlation levels between the different online activities of the students and their final exam grades, as well as for the online content access. Between 28% and 33% of the students which were included in the proactive triggering process were more efficient than students which were not part of the triggering process.

However the initial results obtained in the first paper could not be used for a general conclusion but they served as a reference to compare future results.

The second paper analyzed three studies developed at the University of Luxembourg inside the local LMS environment. The first study, also described above, was included in this article for comparison purposes. The two other studies were based on the same idea as the first one but were addressing different questions related to proactivity and its effects on the LMS platform.

Students were divided for the second study into two groups – the study group and the control group. The difference between these two groups was that the study group received email notifications to increase their attention towards their online assignments. The notifications were triggered by six proactive rules running at predefined time intervals during the whole semester. Results showed a consistent increase in student activities in the period right after the rules were deployed. In the end the marks of the two groups were compared and the success rate of the students in the study group was with 20% higher than the success rate of the students in the control group. Also the study group obtained a good correlation between the on-line forums and the related exam questions.

And finally the third study was conducted in order to relate the student's activity with the proactive notification system and the final grades. As a slight modification, all the students were involved in this study without having separate groups. But the concept

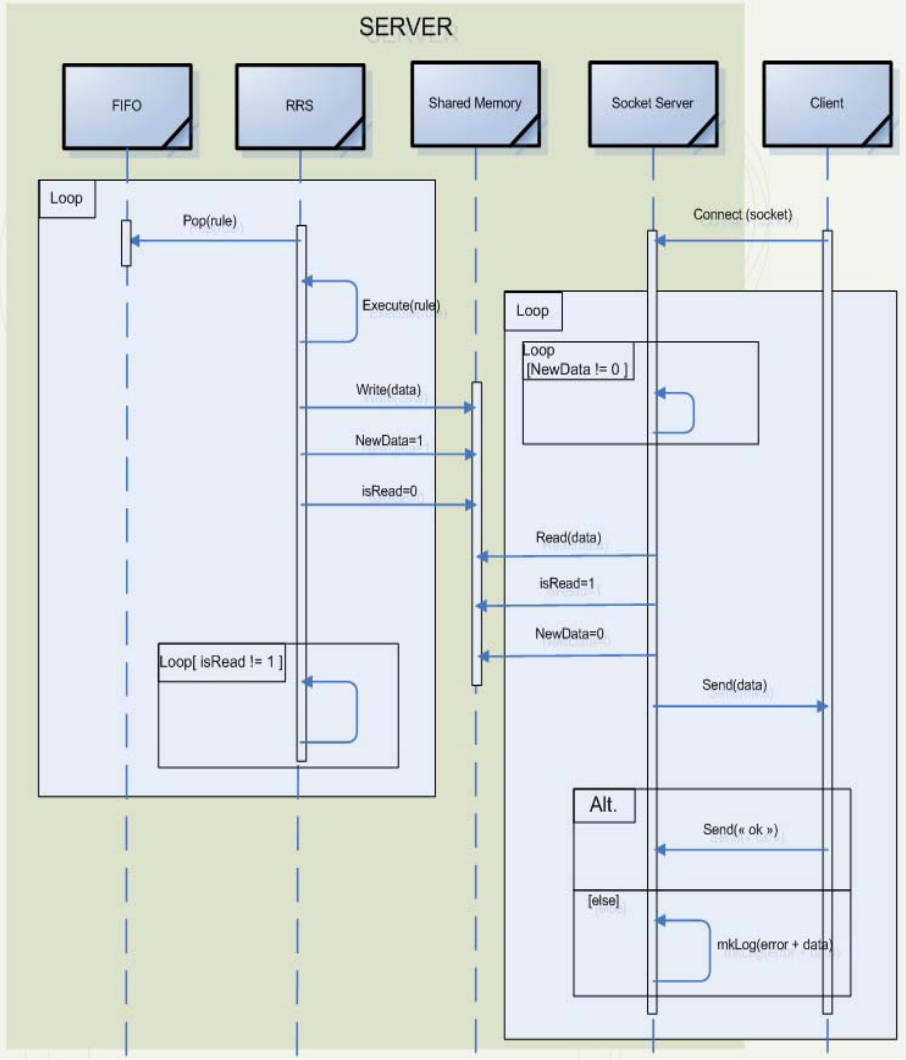

Figure 3.6: Message sending sequence diagram

remained the same – students were notified about their on-line activities like forums, and their level of participation was analyzed after they receive the notification messages. Results showed that students which were more engaged in reading their messages and the forum posts had better marks for their final exam.

Even though these results were quite relevant, a direct relationship could not be yet established between proactivity and the continuous, step-by-steep, learning process. The most important achievement of these papers was the implementation of the proactive system and obtaining positive results for students' activity in a PLMS. Future work, as described in the conclusion of [Coronado and Zampunieris, 2010], would consist in finding patterns for the learning process and developing new proactive rules for these patterns.

Current research in the field of PLMS is done by Denis Shirnin as a PhD thesis at the University of Luxembourg under the supervision of professor Zampunieris. As mentioned in his PhD thesis description [Shirnin, 2012], published at the University of Luxembourg as an internal document, he will focus on researching, analyzing, testing and validating results obtained from student interaction with the on-line learning platform, in order to demonstrate the full capacity and potential of such an proactive system.

3.3 Communities of Practice (CoPs)

At the beginning, when the idea of CoPs was created, the research community was more interested towards creating the theoretical foundations for CoPs rather than developing software systems that would integrate the latest technology.

Nowadays, CoPs represent a crucial research field because of its exceptional results in creating groups for people who have a common goal in a particular domain. Even though it is quite an old topic and many papers were written, this thesis focuses mainly on the work of the famous anthropologist Etienne Wenger. He is well known in the international research community for formulating the theory of situated cognition and more recently for his work in the area of CoPs.

As people come together in close groups by their own will, the success of such collectives is determined over time by their aptitude to create plenty of excitement, attention and knowledge to attract the interest of their members. A community can be inspired by lots of external and internal factors like a serious issue or a disagreement cause by a new topic.

A very intriguing aspect of bringing people together in social groups is what actions members of a certain community could do. At the top of the list would be actions like sharing information, advices, insights, hints and tips, explaining their tasks, helping and consulting with other people, discussing multiple approaches to various topics, their aspirations or their needs, solving occurring problems, exploring common research fields, creating pattern, standards and resources, organizing and maintaining their own documents, understanding and developing new and better perspectives, approaches and practices. This set of actions was firstly explained by Wenger in his first book [Wenger, 1998] where his primary goal was to introduce the concept of community of practice.

Wenger and Snyder explain in [Wenger and Snyder, 2000] why communities of practice are structured differently than other forms of organization like teams, working groups and informal networks. According to them, these differences appear in terms of purpose (i.e. what were they designated for), cycle of life (i.e. the period of time they were designated to last), membership (i.e. who is allowed to be part of such groups) and structure (i.e. which are the reasons for a group to stay together).

While participants inside communities are selecting themselves, people are becoming members in a formal working group by reporting to the group's supervisor, in a project team by being assigned by a manager and in an informal network by associates, friends and acquaintances. Another aspect is that communities survive as long as there topics to be discussed, ideas to be shared and resources to be used. In another order of ideas, while the interest of a group's members is taken care of and does not stop, there is no chance of putting an end to a community. As for the other forms of organizing people, they either last until the following reorganization, in the case of formal working groups, or until the tasks have been completed, in the case of a project team, or as long as people are motivated for connecting, in the case of an informal network. In terms purpose, they can differ because communities try to build, improve and share knowledge, while project-teams try to finish only their tasks, informal networks try to gather and distribute business information and formal work groups work on delivering products and services.

3.3.1 Benefits of using CoPs

In their book [Wenger et al., 2002] published in 2002 , Wenger, McDermott and Snyder agree that while communities are naturally created, institutions and corporations from all over the world have to become more engaged and proactive about cultivating and

integrating them into their structure. Practical forms and methods are given in the book for directing networks of people to reach their maximum level of usefulness without losing their primary goals, which makes them so important.

The authors also offer different classifications of advantages offered by communities of practice. Benefits can be obtained immediately (*short-term*) or after a longer period of time (*long-term*), and they can be seen from multiple points of view: at the level of each member (*small-scale*) or at the level of the entire organization (*big-scale*).

While short-term gains of a corporation include reduced time and costs spent on searches for specific information, multiple perspectives on various topics, a developed environment for problem solving and augmented quality of decisions, the long-term ones are more valuable as they contain aspects like finishing well-determined strategic plans, the ability to innovate, make use of emerging domains, conserving knowledge and anticipating technological findings.

Aliveness is identified as being a characteristic which defines a collective of people, as it cannot be controlled, manipulated or imitated. Even so, the authors argue that aliveness is not certain to appear automatically. There are countless examples of communities that did not evolve beyond the status of a team because they either failed to attract a minimum number of participants or the common goals were not well defined. In most of these situations the failure appeared in an incipient phase as they were not able maintain themselves. Opposite to formal work-groups and project teams, communities survive on the interaction of their members. Interaction can be provoked by many means, either by the system itself, by users or by special events that occur regularly.

A more recent study [Probst and Borzillo, 2008], which tried to establish the reasons behind failures while building communities of practice, was conducted with the help of more than 50 CoP leaders from big corporations like **IBM**, **Mazda**, **Siemens**, the **United Nations** and **CERN** [1] . The lack of a core group, the rigidity of competences, a low level of cooperation between members, the lack of identification inside groups and practice intangibility were indicated as the main motifs for the lack of success when using CoPs. Based on these findings, a solution of having a governance committee is proposed. This will lead to new favorable circumstances for inter-CoP exchange of beneficial practices, for increasing the visibility of each CoP to the upper management layers, for uniting and merging multiple CoPs, and to manage activities inside CoPs. Suggestions are made for a closer examination of CoPs at the level of each member to gain a profound understanding of the majority of factors causing success and failure.

There are a couple of steps to be followed in order to cultivate a successful community of practice. According to Wenger [Wenger and Snyder, 2000], this mainly depends on the aim and objectives of the collective as well as what resources are available and to what members are attracted inside these groups.

The first step includes designing the community to progress in a natural way as anticipated. Because they have a dynamic structure, where factors like interests, goals or members could change, communities should integrate tools that will easily allow these changes to take place.

Then, opportunities should be created for an open dialog between the internal and the external environments. A good community design would benefit a lot from the insider's perspective as they know what is going in the heart of a collective and from the outsider's point of view as they are aware of all the new information and opportunities.

Allowing multiple levels of participation would increase the activity of the members and the group's popularity. Wenger acknowledges three main levels of participation

[1] European Organization for Nuclear Research.

[Wenger and Snyder, 2000]:

- ***The core group***, which is the most important layer, who is all the time involved in discussions, debates, activities, tasks and projects. It is represented by a small group of people that identify new topics to address, brings the community closer to its learning schedule and which takes on guidance roles inside the group.

- ***The regular group*** which are active from time to time but not at the same point as the people in the core group.

- ***The peripheral group*** that is more or less involved in the learning process but which never, or rarely, contribute with something. They are not that passive, speaking a lot in private about the topics being discussed publicly. They represent, in many cases, the majority in a community.

Public and private places have to be established to give an opportunity to members to flourish. Private spaces for discussing and exploring thoughts are important for people that don not have the courage to express their opinion publicly. A manager would moderate connections between participants and available resources in a transparent and individualized way, specific to everyone's own needs.

The fifth step describes what a community should do to keep focusing on improving and sharing its value. Communities of practice should include the idea of explicitly showing, through discussions, the value and productivity for each of its participants.

A collective of people should be able to mix familiarity and enthusiasm. As learning opportunities are a part of communities of practice, members should be able to form their experience together with other members by examining and analyzing knowledge connected to their subject.

And finally, a rhythm has to be conceived and followed by the community in order to achieve its purposes. A natural cycle of events and activities has to be maintained regularly inside the group as they give participants the possibility of come together, think and evolve. The frequency of events or the pace should be maintained at a medium level by the member's commitment. Evolving to fast is to be avoided as it can become overwhelming in some groups.

3.3.2 Online Communities of Practice (OCoPs)

According to Wikipedia [Wikipedia, 2012b], an Online Community of Practice (OCoP), sometimes referred to as Virtual Community of Practice (VCoP), is a particular organizational form which runs on knowledge, while being maintained using the Internet. Its purpose is to stimulate knowledge sharing, learning, and change.

An on-line community needs to have a couple of more general characteristics of CoP including actively engaged users or specialists, in the case of very specific domains of interest, that are taking part and regularly contributing to processes like collective learning [Wenger, 2006]. As an extra element, social groups have to be defined within these communities in order to benefit from creating new ideas and distributing the other people's understandings. The context where the knowledge is exchanged plays a very essential role. Normally, people that will have hard times to meet physically can use a technological platform as a place for solving problems related to their domain of interest. Finally, multiple dimensions must facilitate the long-term management of support as well as enable immediate synchronous interactions.

In one of Lithium's company reports [2012, 2012], which provide practices and tips to reduce expenses and to increase the user's overall level of achievement, on-line communities are places where a collective of people is working conjointly for reaching the same goal. The start point is given by tools like forums, chats and discussion board. One major point they underline is that advanced search tools, that incorporate only the essence and meaningful knowledge, should exist inside these social group.

In Etienne Wenger's survey on community-oriented technologies [Wenger, 2001], a section is dedicated for identifying facilities useful for on-line collectives of people and for describing which properties should an ideal platform have for supporting communities of practice. The system should be:

- Intuitive, easy to handle and fast to learn;
- Open for a fast integration with other software platforms used by members of that community in their daily work. Being compatible with other software will increase participation of all members in only a couple of extra steps taken for the integration phase;
- Supports a big number of users simultaneously;
- Not so expensive and possibly open-source as countless communities start realizing only later the importance of their outcome;
- Customizable, with various levels of customization in order to fit the needs of both managers and software developers;

A more complete list of features is given in section 4.1 of chapter 4 - Theoretical Background.

Online communities of practice are grouped in multiple categories including social networking sites (e.g. Linkedln, Facebook, MySpace and YouTube), virtual worlds (e.g. Second Life and WhyVille), tools for educators (e.g. Classroom 2.0, Teacher Leaders Network Forum, Educator's PLN, eMSS and ETLO) and information sharing platforms (e.g. Wikis, Google Docs, Blogs).

Specific tools are a key element in communities of practice as they are the instruments which members operate. According to Wenger these tools should include virtual facilities like: a main page where topics and domains are shown and described, on-line private and public place for speaking about different subjects, a gadget for sending questions to the whole community or just to a small group of people, a management committee with specific information about their areas of expertise, a common workspace where synchronous activities could be performed, a document repository for a better knowledge management, an advanced search engine for fast data retrieval, a set of management tools for administrators for monitoring, managing and controlling internal activities, including which resource is being downloaded, who is more engaged in the group's routine, and finally, ways of distributing members to smaller collectives, subgroups and project team should be available.

3.4 Situation and Context Awareness

SA is one of the most crucial elements for evaluating human-machine interaction in many areas such as aviation (e.g. inside the cockpit of an airplane or inside an air traffic control room), the industrial sector (e.g. regarding power plants) or even in the army (e.g. ship navigation). Lots of researches focus on measuring SA in precise contexts, at a specific

moment in time. For example, a plane's pilot needs to be aware of many complicated and changing situations that may appear , or a monitoring system of a nuclear power plant detects unusual events and starts taking decisions until people are made aware of the occurred events. SA is highly necessary for the decision-making process because it is important to have the full picture of what is going on for taking the best decision in each situation.

In [Schilit et al., 1994b] the authors focus more on analyzing software that is examining and reacting to the context change of individuals. They argue that such software can boost people's interactions with computers, devices and other individuals, while providing assistance for discovering unfamiliar locations. They identify four categories of context-aware applications: automatic contextual reconfiguration, proximate selection, contextual information and commands, and context-triggered actions. W.Maalej extends this list in 2011 [Maalej, 2011] with areas like: knowledge sharing, information allocation, awareness creation, traceability and tool integration, and personal productivity management.

In their article [Shu and Furuta, 2005] , Shu, Kazuo and Furuta argue that previous models of Team Situation Awareness (TSA), which is a major contributing factor in forming relations between members that are engaged in cooperative activities, are not sufficient or suitable for studying complementary team processes. They specially refer to those models where TSA was introduced as the intersection of SA belonging to each particular member. A suggestion is made for integrating Individual Situation Awareness (ISA) into the cooperative activity of a team for having a complete overview of TSA. Their new definition of TSA contains mutual beliefs and ISA at three consecutive stages that include: feeling the surrounding elements, understanding the actual situation and assessing future events. The first level is about looking and perceiving basic information, the second one about thinking and understanding the multiple meanings of information and the last level is the place where these meaning are applied for anticipating better what will happen.

According to Jakobson et al. [2007], the Situation Management Domain incorporates SA together with Situation Semantics, Calculus and Control. But despite lots of merits attributed individually to these disciplines, caused by a big volume of research in situational aspects and behaviors of humans, systems and corporations, a symbiosis was not yet reached. The authors redefine the concepts of Situation Management (SM) , while identifying and analyzing several related technologies. They see SM as a structure of concept, ideas, models and precise technologies for distinguishing, observing, perceiving and predicting situations or events that might appear in dynamic systems during their life cycle.

A system has the property of being aware if it exploits different context to provide accurate information and services to its users, when relevancy counts on what the actions of the user . There are some issues with context adaptation due to the fact that it cannot be easily recognized, measured or evaluated. Locating the user, in the case where he/she is not in front of a computer, has become a simple task thanks to very advanced positioning systems. Until now, context-awareness has mainly been analyzed from the technical perspective and the studies have focused more on locations. Multiple experiments have been done with software systems but only a couple of small-scale results are available for a proper comparison [Kaasinen, 2003].

Having a system evolving naturally, effective and efficient implies the fact of being aware of changes. To modify a system, a first step has to be taken in understanding how it works and what are the consequences provoked by changes. Since change is unavoidable and causes systems to drift from their initial purposes, the most constant and valid source

of information still remains the system itself. In [Oscar Nierstrasz and Rothlisberger, 2008] the authors describe how would a "self-aware eternal software" support its own evolution. The list includes steps like providing accurate models for changes, history of events and logs at a platform's level, analyzing permanently the static and dynamic evolution of different aspects for tracking new properties, and narrowing the gap between the point of view of the developer and the domain model.

Chapter 4

Theoretical Background

4.1 The Ideal Proactive System for CoPs

Companies and organizations all around the world are faced with picking the right technology for supporting and developing communities. Community development refers to the process of enabling and encouraging people to actively engage in obtaining the desired outcome of their work. Proactivity makes CoPs even more powerful, granting them with an appropriate level of support, help and guidance from the system itself.

This chapter is mainly dedicated for identifying characteristics which are supporting the idea that using proactivity in the context of communities of practice will enhance the learning process, as way of cultivating a profound knowledge, and the environment in which people work.

The ideal system needs to answer a couple of important questions like:

- Does it need to contain specific requirements from the users?
- Does it need to respect a special structure?
- What are the major features required to have successful communities of people?

Establishing a theoretical model is very important because it can be used as a reference to see how compatible a software system could be with concepts like communities of practice and proactivity.

First of all, the majority of current software platforms on the market need to use groups of people which work together for achieving common goals. Normally, this kind of environments are quite static and only respond to the user's commands , totally opposed to proactivity which supposes that a system also works on behalf of the user. The proactive engine is conceived to work aside another system but there are special cases where systems are using certain aspects of proactivity and just need to integrate others characteristics in order to have a fully dynamic and user-dedicated structure.

Checking the characteristics of the ideal proactive system is quite important in determining if a system needs enhancements like proactivity and CoPs. Of course, the checking step should include the process of identifying the common goals of both systems. And finally, an analysis should decide how compatible are the two platforms being compared.

Two main directions of describing specific features are taken in the following subchapters. The first one explores more the attributes of a proactive system. The second one, more conclusive from the regular user's point of view, contains major requirements for creating a good on-line environment in order to apply CoPs.

In general, issues like reconfiguration, maintenance, troubleshooting and decision-taking tasks bring with them other problems like high costs and time-consuming solutions. The cause of these problems is often due to the lack of functionalities. Therefore, there is an increasing demand on the market for systems which are robust, low in complexity, while providing automated processes for achieving the desired requirements within an equitable context.

4.1.1 Key attributes of the ideal proactive system

Key characteristics	Description of the characteristic
Context awareness	The process of being aware of what is happening around to understand how information, situations, and user's actions will affect general objectives, immediately as well as in the near future.
Activity/non-activity detection	Used to detect intense activity from users or in the opposite case, the lack of actions from certain users.
Dynamic	At any time, the dynamic system could change its structure, status or any of its components.
Self-adaptive	This feature aims to modify attributes or resources in response to various changes in the environment. It is closely connected to context awareness and proactivity.
Extended monitoring	Can be defined as an observation set of tools which help to check and verify user actions, system status, existing errors and if activities are done in a correct, efficient and dynamic way.
User assistance	An efficient and automated way for guiding and providing useful information and hints to the user.
Advanced notification tools	Designated for notifying users of particular events or activities. Alerting administrators of suspicious activities or events, while saving precious response time.
Reporting initiative	Generates reports and statistics, based on thorough analysis, for improving the overall performance.
Intention prediction mechanisms	Predicting user's intentions based on his/her actions by using various detection techniques.
Initiative-taking	Assumes an active role in numerous situations where the user's explicit action and intention is missing.
Feedback and rating mechanisms	A crucial part of any software, where users can comment on, rate and contribute to the content, while expressing their opinion publicly.
Advanced user management	Multiple categories of user roles, with different permissions. The system needs to automatically differentiate between learners, teachers and administrators, and their specific responsibilities.

Table 4.1: Key characteristics of the ideal proactive software system

In the table immediately above, key characteristics of a system that wants to be

proactive are identified and then, each characteristic is briefly explained. These capabilities have an essential role as they try to solve some of the flaws of current complex systems, which have many distributed applications. One example would be the explicit human supervision needed to continue certain processes and operations at any time, in all the possible conditions.

4.1.2 Specific requirements of a community-oriented system

Specific requirements	Description of the characteristic
Minimal complexity	Accessible for users which do not possess great technical skills and cannot operate in such groups. The lack of expertise of many people prevents them in using web collaboration tools.
Ready to serve user's needs	The community should help the user in his daily tasks in a transparent way, without becoming a real burden.
Extensibility	Easy to extend by adding or removing elements, i.e. plug-ins, from the structure of the social groups, without disturbing the already running processes.
Training methods	Very useful for newcomers. This responsibility should be divided between the system and the community, both providing useful information to the user.
Accessibility	Users should be able to access off-line resources if they are not connected to the Internet. Documents should be located in a cloud, not stored on a single server.
Resource handling	The members of the different communities should be able to manipulate, optimize, prepare, finalize and advise on how to use the system's resources.
Advanced communication tools	Inside their groups, interaction between members should be supported with lots of synchronous and asynchronous ways of exchanging information.
Adaptable	The groups of people should not have a limited number of members or resources, no matter how large is the community.
Governance and decision making	CoPs need a manager in each community, responsible for selecting the best configuration, granting access roles to the learners, choosing the best gadgets to have inside the group and finally, to solve general issues.
Focused on practice and knowledge	Develops, attests and broadcasts specific practices. Enhances, organizes and shares the daily knowledge.
Capturing Experience	Retaining the experience of users inside the community so that it will not rely on the knowledge of one member.
Customizability	The user should be free to customize her/his own working environment, to make it more familiar.

Table 4.2: Specific requirements of community-oriented system

The table contains more specific features, necessary for defining how would a strong community look like and what kind of actions will it be capable of supporting. These

specifications are required especially for online CoPs, where users must actively participate in a process of collective learning within their specific domain of interest.

4.1.3 The perfect system

Trying to describe the perfect software system for a specific domain is relatively hard. Let's take for example the case of online LMS systems. Even though there is quite a big variety of web based learning systems, a perfect system wasn't yet identified because the needs of communities of people are different, given certain specific requirements. Sometimes these differences are not so obvious (i.e. when a new community searches for online platform to organize its resources and they just need a basic system to fulfill their needs) they take the example of other communities which have tried out similar software, but in the majority of the cases they just do not match.

Following the characteristics of the models mentioned in the tables above and to answer the questions mentioned in the beginning of this chapter a concrete example is provided below of how would an ideal LMS would look like. It actually contains an analysis of the minimum vital requirements which would help the user in his/her daily routine of ameliorating and sharing their knowledge.

So, how would a ***proactive LMS which supports communities of practice*** look like?

To start with, the CoPs inside the LMS would be goal-oriented, especially for learning purposes. Interaction between learners or between learners and instructors would represent one of the biggest concerns of such environments. Ensuring support needed for intense collaboration, the LMS should facilitate as much as possible the work and activity of each person. It has to be intuitive for the majority of the untrained users and easy to use when they want to navigate and find something they need. CoPs should be customized to each group's desire, based on a list containing specific parameters.

Members of each community can often be in locations where a stable Internet connection could not be maintained, so off-line materials and resources should still be available to these users. Speaking about resources, a central repository for on-line documents, organized as a library, would be appreciated. An extra improvement would be the system's compatibility with mobile platforms, where the accounts and shared resources should be synchronized. A good LMS shouldn't simply be just a way of delivering different materials to students. It should be a place where students can construct their own learning style and improve the way of understanding various courses. Comments from users should be available to other learners as they might be lost or stuck in their activities.

Means of communication would represent a crucial part of the whole system. Multiple tools should be offered to users for exchanging information both in synchronous way and in an asynchronous way. Verbal ways of transmitting information to the user would be considered as a big plus. Visual support should be offered to the user through a graphical user interface. The GUI should be flexible and adaptable to both to the instructor's and user's needs. As instructors are quite busy persons nowadays, the LMS should have fast and easy ways of using authoring tools.

Different system roles should be applicable to users, depending on their goals and tasks. Permissions should be clearly defined in order to have a well-defined hierarchy inside the community. Features like reporting, tracking and recording should not be missing from such a platform. The firsts to benefit from these elements should be the administrators, closely followed by the teachers and finally the learners themselves.

Out of the many exiting LMS, Moodle™ was chosen because it is widely adopted in many academical circles and it represents a viable option for other corporations. Maybe it is not the closest platform to the above example but the lack of certain system features offered the possibility of deploying the proactive engine and ameliorating the whole learning environment. In section 4.4 of chapter 4 - Theoretical Background, personal reasons for picking Moodle are explained, together with its more general motifs which are listed as pros and cons.

4.1.4 Advantages of using Proactivity and OCoP

The advantages of using Proactivity and OCoPs clearly outweigh the disadvantages, as they represent, without any doubt, a real solution for lots of software systems. Identifying and highlighting some of the major benefits of using such concepts are quite important for companies and organizations that wish to integrate new technologies.

They can be grouped at different levels: benefits from the point of view of the entire organization or from the point of view of the user. These approaches are very significant and closely interconnected.

As they can be also separately incorporated into new or existing systems, advantages of each field of research are threated apart. So, what are the **benefits of a proactive software system** over a reactive and static system? The list contains the following characteristics:

- Identifying opportunities for improving the whole environment by adding resources and by changing the state of the system;
- Prevents problems and special situations by using a proactive analysis;
- Anticipates the user's need and intentions and makes decisions which will help the user in obtaining his/her goals;
- Provides help, assistance and guidance to users when they are in trouble or when they do not know how to continue their activities and assignments;
- Real-time event detection for unpredicted events;
- Self-adapting structure-wise and output-wise to meet the changing needs of its users;

For Communities of Practice, lots of articles and papers were written to emphasize their advantages. One of the most relevant works was done by Gannon and Fontainha Gannon-Leary and Fontainha [2007]. Among those benefits, the most important are:

- **Enhanced learning environment** – Gives the user flexibility and creativity, while avoiding duplication of existing work;
- **Innovative** – Great possibility to express for persons who have issues in communicating verbally;
- **Fast Familiarization** – Users get familiarized much faster to the environment if they have to used it for multiple tasks and activities;
- **Knowledge sharing and learning** – A common place for sharing and developing new ideas;
- **Time saving** – Accessible without physical presence;

- **Relationship building** – Breaking social barrier between people, because of languages, clothes and appearance;
- **Active participation** – People are encourages to actively participate in activities and interact with each other;
- **Dynamic Structure** – Communities are not limited at the level of their structure and they can be quite adaptable, depending on the needs of each group of people;

There are lots of firms and organizations which provide a proper infrastructure /environment for developing CoPs and for using proactivity to make their staff more productive, engaged and focused, in order to lower costs and increase the income.

4.2 Solutions to have an ideal system

In the previous chapter, the characteristics of an ideal platform are listed for applying proactivity for communities of practice. The local Moodle platform at the University of Luxembourg was lacking local communities and social interaction inside these groups, so a solution needed to be found in order to improve the on-line social life of the students.

The content of this chapter is dedicated for finding a proper solution to solve the problem of missing organized and focused groups of students, which have something in common like the city of residence or the study formation they actually follow.

A moodle course, called *Social Groups*, is created (*manually in the first phase*) and then moodle groups of users are defined inside this course. The routine called "Initial creation of the groups" was the first step in designing how to create and manage groups of students in Moodle. This routine is described in details below, in section - 4.3 . Its structure is decomposed in three main stages or categories of rules that are later explained separately.

Basically, the whole idea of having a cycle that would repeat actions that respect a certain structure came from the need of having rules that run continuously with the possibility to adapt to new circumstances triggered by user actions or even by non-actions.

Groups can be created on other initial criteria but for testing purposes the current hometown of the users makes more sense to begin with, in the context of communities of practice. Other social groups will be created later on, based on the different user's formation, e.g. "MICS", "BPINFO". These communities of practice are created to bring users together to share information which normally could not be obtained inside Moodle. All these mechanism of creating social groups are analyzed in advance and only the most relevant ideas come through to be finally implemented. They, however, are to be integrated in the proactive cycle illustrated in figure 4.1.

Ideally, the system should be intelligent enough to adapt to certain situations like creating and managing groups based on the needs of the majority of users or at least to provide an advanced mechanism of notifications which would at least allow users to get information without any explicit actions. Moodle can adapt very fast to the needs of different communities, but it is still a static system. All the modifications done to the system need to be made by users, with explicit command. The rule engine of the proactive system is designed in such a way that multiple routines can run simultaneously without affecting the operations of the other routines. In fact, this concept of the proactive cycle with different rules and meta-scenarios belongs to a bigger idea of having a very intense and useful social interaction inside a community using a certain software platform. Even though the example below is just a proof of concept, it is ready to be deployed on multiple

systems which are focused on other specific domains and lack the part of social interaction inside their platform.

4.2.1 Important specifications

Only users with Moodle *student roles* will be involved in the social groups. Professors and assistants will not be asked to join social groups in the beginning because students would be more reluctant in exchanging information, such as previous exams of a certain course.

Rules and meta-scenarios can be distinguished quite easy. Their names either start either with **Meta-Scenario (MTA)**, **S** for scenarios or with **R** for simple rules. Then numbers composed of three digits are assigned to them. The first digit indicated to which stage of the cycle the rule belongs to and the second and the third digit are just an unique identification number. They indicate if the rules or scenarios are part of a bigger group of rules which will be executed, as part of the same sequence. So, the maximum number of scenarios, rules or meta-scenarios that can be executed in a certain sequence is reduced to 99 because of the last two digits.

For example after rule R201 is executed, the rule R202 will follow in the execution queue because it is triggered by R201. There are a lot of rules that belong to different stages that are executed in parallel and this is very important in showing that his whole proactive cycle is dynamic, not a sequential one. More details about executing rules in parallel are showed in figure 4.3. Even though, in the practical implementation stage, scenarios, meta-scenarios and rules keep the same programming structure, they are differentiated at a far more important level (i.e. at a *conceptual level*).

The Proactive System (PS) is a goal-oriented mechanism with a clear objective to execute sets of rules for improving the activity of the whole system and for helping users in achieving their purpose. Rules are considered a basic structure, used to make some actions like sending notifications to the users or registering in the database useful information and generate other rules. Scenarios are more complicated structures as they have different missions. They can be distinguished by their set of features, complexity and area of application. And, with the area of application comes the third category of rules, also called meta-scenarios. These are groups of scenarios which have similar characteristics and purposes.

For example, scenarios and meta-scenarios are needed for changing the state of the system. This means lots of structural modifications will take place with the creation of many resources.

One more thing worth to be mentioned is that scenarios and meta-scenarios are **time-determined mechanism**. They tend to run as long as the proactive engine is working, at a predefined interval of time. Normally, these time intervals are relevant for the type of job they have to accomplish. For example, there are scenarios which run each week for checking if there are students who completed their city field name, which initially was left empty, or meta-scenarios which run once or twice per semester. The last ones check all the social groups for members that do not want to be part of these groups and propose new communities.

4.3 Initial creation of the groups

All scenarios, meta-scenarios and rules are part of one stage of the proactive cycle, except the first meta-scenario which triggers the whole mechanism. This cycle will set up social groups for moodle users, based on their city of origin. The participants of the firstly

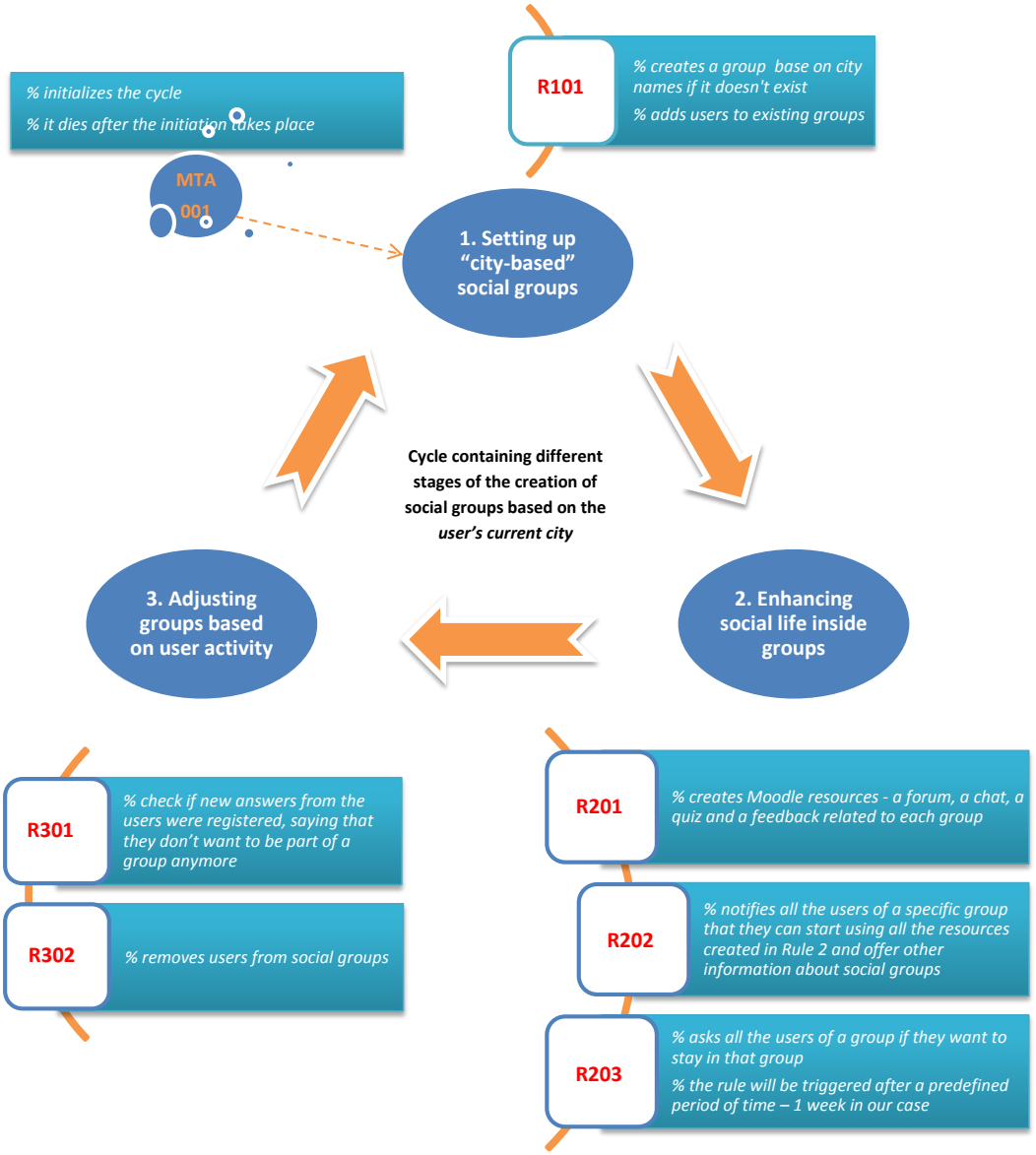

Figure 4.1: Initial creation of groups

created communities will have in common the city of origin which is taken from their description that they have completed on their moodle profile.

The role of the first scenario is to trigger the whole cycle shown in figure 4.1. After its initiation, the cycle will run until the rule engine is stopped. The first steep would be to inscribe users in social groups without being asked before if they want to join or not. This question will be asked later on for every participant. The main idea behind this reasoning is to make users curious and more involved by skipping some extra steeps used for the creation of the groups.

The idea is to have all the users, which come from the same city, in one place where they could interact and exchange precious information such as:

- Courses, exams, pdf files, etc. ;
- Create new friends that could help with transportation or with other information about accommodation, sports, books, etc. ;

- Check for news related to student jobs, student activities and events;

Because the city of origin is a text field in the moodle description of each user, special cases have to be taken into account:

- *Luxembourg, Luxembourg-Ville, Luxembourg City* ;
- *Esch, Esch/Alzette, Esch-sur-Alzette, etc.* ;
- *Neighborhoods of Luxembourg City : Kirchberg, Limpertsberg, etc.* ;

The second part of the proactive cycle is very important because it represents the reason why social groups are created in the first place. The first rule (i.e. R201) creates groups and specific resources related to this groups, which are nothing else than means of a better interaction between the users. Forums, quizzes, documentation available on pdf files and questionnaires contain specific information related to each group.

R201 triggers the second rule of this stage, R202, which informs users that they are part of a community. A generic message is sent to all users immediately after the rule R201 has finished. The third and last part of the proactive cycle is taking specific care of the users, in the sense that the proactive system is modifying by itself the social groups, without any specific commands from users.

In fact, the system checks, with the help Rule 305, if users still want to be part of the initial social groups. Based on this knowledge the groups are adjusted in such a way that only people who want to really remain in the group and are very active stay together. As for the less active users, they are encouraged to become more engaged and to interact more inside the local Moodle communities. Ending the last stage of the initial phase of creating groups means that new groups can be created on other criteria like the study program where the students are currently inscribed in.

4.3.1 Initial Scenario

Rule ID:
S001
Rule Description:
Initiates the proactive cycle which creates the first social communities based on the city name completed by users in their description. It is supposed to run only once, dying after triggering the proactive cycle.
Parameters:
long executionTime *% unix timestamp time*
Data Acquisition:
string cities [] = getDifferentCitiesFromDB(); *% retrieves all the different cities from the DB*
boolean wasExecuted = checkQueueIfTheRuleWasInitiated();
Activation Guards:
wasExecuted == false *% checks that the rule was not created before*
Conditions:
return true
Actions:
no action
Rules Generation:

```
if (activationGuards)
    foreach city in cities[]
        if(getNumberOfUsers(city) >= 3 && checkDbIfGroupAlreadyExists(city))
            createRuleR101(city)
        end if
    end foreach
end if
```

Notes:
- *Special cases have to be considered for these fields:*
  - {“Letzeburg”,“Luxembourg-Ville”,“Luxembourg Ville”, “Kirchberg”, “Limpertsberg”, “Luxembourg City”, “Luxemburg”} belong to Luxembourg Group
  - {“Esch/Alzette”, “ Esch-sur-Alzette ”, “ Esch Alzette ”, “Esch sur Alzette”} belong to Esch-sur-Alzette Group
  - {“Limpertsberg”, “Kirchberg”} belong to Luxembourg Group

4.3.2 Setting up “city-based” social groups

Rule ID:
R101
Rule Description:
Creates a new social group based on the city name, which is received as a parameter. It also gets all the users (which have the same city as the group name) from the database and then inscribes them to this group.
Parameters:
string city *% the group’s name will be given after the city parameter which is received as a parameter, when this rule is created*
Data Acquisition:
string groupName = city;
int users[] = getUsersWhichComeFromSameCity (city);
Activation Guards:
return groupExists();
Conditions:
return true;
Actions:

```
if ( ! activation())
    createGroup (groupName);
foreach user in users[]
    if (userIsNotPartOfGroup(groupID, user.ID))
        inscribeUserInTheGroup(groupID, user.ID);
    end if
end foreach
```

Rules Generation:

```
if (activation())
    createRule201 (groupName)
die();
```

4.3.3 Enhancing social life inside groups

Rule ID:
R201
Rule Description:
Creates the necessary means for a better social interaction among users, depending on the available resources. These resources include a forum where the users can discuss different topics, a live chat, a quiz and a feedback containing questions about the idea of using social groups in Moodle's context.
Parameters:
groupName *% contains a certain group name*
Data Acquisition:
description = " Welcome to the group of people that are living in the same city. You can share resources, speak with the other users and make use of all the available resources on this group."
Activation Guards:
return true;
Conditions:
return true;
Actions:
setGroupDescription(getGroup(groupName), description);
createMoodleForum(groupName);
createMoodleQuiz(groupName);
createMoodleFeedback(groupName);
createMoodleChat(groupName);
Rules Generation:
createRule202 (groupID)

Rule ID:
R202
Rule Description:
Used to inform users that they are part of a group and that they can start using this group for sharing and developing their knowledge. Send a generic message to all the users of a social group. One rule is created for each social group.
Parameters:
groupName *% contains a certain group name*
Data Acquisition:
users[] = getUsersOfSameGroup (groupName);
messageID = getMessageID (rule.ID); *% recovers the message ID related to this rule from the database*
message = getMessageText (messageID); *% gets the text of the message based on the ID % = "This message was sent just to inform you that you were inscribed in a new social group. You are part of the group of people which have the same origin city as you and are inscribed at the same faculty."*
Activation Guards:
return true;
Conditions:
return true;
Actions:
foreach user in users[]
if (userDidn'tReceiveAlreadyTheMessage (user.ID, message))
sendMessage (user.ID, message);
end if
end foreach
Rules Generation:
createRule203 (groupID)

```
if (checkQueueIfRuleWasn'tAlreadyCreated("MTA311"))
        createRule311(getSystemCurrentTime());
```

Rule ID:

R203

Rule Description:

Asks the users of a group if they still want to be part of a certain group. This rule will be activated after a small period of time (2 weeks) where users had enough time to get involved in the activities of a group.

Parameters:

```
groupName                                   % contains a certain group name
```

Data Acquisition:

```
questionID = getQuestionID (rule.ID);       % recovers the question ID related to this rule
                                              from the  database
question = getQuestionText (questionID) ;   % gets the text of a question based on the ID
                                            % = "Do you still want to part of this group?"
timestamp = getGroupCreationTimestamp (groupName); % when the group was created
users[] = getUsersOfSameGroup (groupName);
```

Activation Guards:

```
if (getCurrentTime() – timestamp >= 1209600)    % 1209600 in unix time represents 2 weeks
        return true;
```

Conditions:

```
return true;
```

Actions:

```
foreach user in users[]
        if (userWasn'tAlreadyAsked (user.ID, questionID))
                askQuestion (userID, question);
        end if
end foreach
```

Rules Generation:

```
if (activationGuards)
        createRule301 (groupID);
        if (checkQueueIfRuleWasn'tAlreadyCreated("MTA321"))
                createRule321(getSystemCurrentTime());
else
        cloneRule203 (groupName);
```

4.3.4 Adjusting groups based on user activity

Rule ID:
R301
Rule Description:
Checks for user answers, if there are new requests to leave a group. If they do want to stay, the question will not be asked again to the same user, letting him/her the option to leave the group by clicking a button. If they don't want to stay they will be removed from the group and their answer registered in the database. This rule is generic, meaning that it will run until the rule engine is stopped. It will generate a rule for each user that want to leave a social group.
Parameters:
lastExecutionTime *% unix timestamp time*
Data Acquisition:
currentTime = getSystemTime(); *% current system time*
users [] = getUsersWhoResponded("No"); *% gets all the users which said they don't want to be part of a certain group*
Activation Guards:
currentTime > = lastExecutionTime+604800 *% 604800 in unix time represents 1 week*
% used to know if it is time for activation
Conditions:
return true;
Actions:
no actions
Rules Generation:

```
if (activationGuard)
        foreach user in users[]
                int groupsIDs [] = getGroupIDs(user.ID);
                foreach groupID in groupsIDs []
                        createRule302 (groupID, user.ID)
                end foreach
        end foreach
        cloneRuleR301 (currentTime);
else
        cloneRuleR301 (lastExecutionTime);
end if
```

Rule ID:
R302
Rule Description:
Removes a user from a group. It is triggered by the rule R301 and it receives a user ID and a group name, which are used to remove a specific user from a certain group.
Parameters:
groupName *% contains a group name*
userID *% contains the ID of the user that wants to be removed from the group*
Data Acquisition:
groupID = getGroupID(groupName);
Activation Guards:
return true;
Conditions:
getUserWasRemovedFromGroup(groupID, userID) == true
Actions:
removeUserFromGroup (groupID, userID);
Rules Generation:
die();

4.3.5 PAM database tables

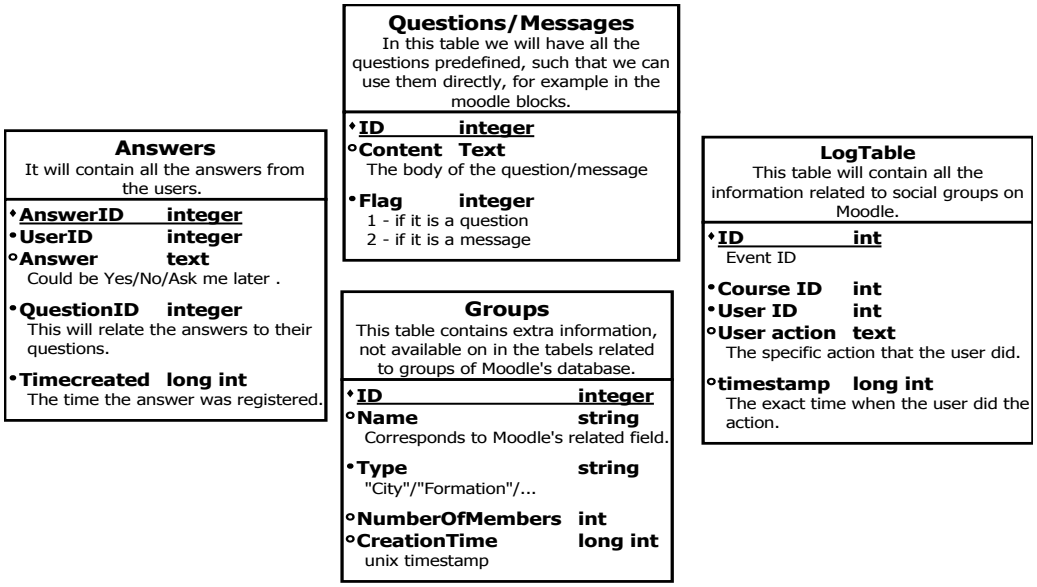

Figure 4.2: Tables stored in PAM

Questions/ Message - This table contains predefined questions and messages that will be sent to the users involved in social groups. The content of these questions and messages is final and can be stored from the beginning in the database because they are generic and don't contain specific information. Because of this users of different groups can receive similar messages, with the same content.

Answers – Connected with the "Questions/ Message", this table stores the answers of each user at the questions being asked in the context of his/her community.

Groups – Can be regarded as an extension of the existing moodle database tables - "moodle_ group" and "moodle_ group_ members". Some fields like "UserID" and "GroupID" contain ID's which correspond to the ones stored on the Moodle system. This extension is necessary because some extra information about groups is necessary and Moodle's database table do not provide it.

Log Table – In this table, all the information related to groups will be stored. Basically it has a similar structure as Moodle's database log table, the only difference being that it is more specific and stores only particular data related to social groups.

4.3.6 Additional scenarios and rules

Rule ID:
MTA311
Rule Description:
Checks for the users that have not completed their current city field and ask them, through a message, to update their information in their Moodle description. This meta-scenario helps users that are not aware of the importance of correctly completing their Moodle description fields.
Parameters:

```
lastExecutionTime                          % unix timestamp time
```

Data Acquisition:

```
currentTime = getSystemTime();             % current system time
users [] = getUsersWithEmptyCityFields();  % gets all the users which have city == ""
messageID = getMessageID (rule.ID);        % recovers the message ID related to this rule
                                             from the  database
message = getMessageText (messageID);      % gets the text of the message based on the ID
                                           % = "Please fill out your city field in your Moodle
                                           description in order to take part in social
                                           groups."
```

Activation Guards:

```
currentTime > = lastExecutionTime+604800   % 604800 in unix time represents 1 week
                                           % used to know if it is time for activation
```

Conditions:

```
return true
```

Actions:

```
foreach user in users[]
        if (userDidn'tReceiveAlreadyTheMessage (user.ID, message))
                sendMessage (user.ID, message);
        end if
end foreach
```

Rules Generation:

```
if (activationGuards)
        if (checkQueueIfRuleWasn'tAlreadyCreated("MTA111"))
                createMTA111(getSystemCurrentTime());
        end if
        cloneRuleMTA001(currentTime)
else
        cloneRuleMTA001(lastExecutionTime)
end if
```

Rule ID:
MTA111
Rule Description:
Searches for users which have changed their current city name because it was empty, and then it inscribes them in the corresponding social groups. It is triggered by meta-scenario MTA311 and it runs once per week, normally after minimum 3 days of the initiation of MTA311.
Parameters:

```
lastExecutionTime                          % unix timestamp time
```

Data Acquisition:

```
users [] = getUsersWhoUpdatedTheirCity();
currentCount = selectUsersWithEmptyCityFieldsFromMoodleDB();
oldCount = getUsersWithEmptyCityFieldsFromOurDB();
```

Activation Guards:

```
users [] = getUsersWhoSaidNoToCityGroups();
```

Conditions:

```
currentCount > oldCount
```

Actions:

```
        setUsersWithEmptyCityFieldsFromOurDB(currentCount);
Rules Generation:
        if (activationGuards)
                foreach city in cities[]
                        if(getNumberOfUsers(city) >= 3 && checkDbIfGroupAlreadyExists(city))
                                createRuleR101(city)
                        end if
                end foreach
                cloneRuleMTA001(currentTime)
        else
                cloneRuleMTA001(lastExecutionTime)
        end if
```

Rule ID:

MTA321

Rule Description:

Checks for users which don't want to stay in certain social groups and then proposes new groups to these users. It runs once per week and it is triggered by rule R203. It belongs to the third stage of the proactive cycle.

Parameters:

```
lastExecutionTime                       % unix timestamp time
```

Data Acquisition:

```
currentTime = getSystemTime();          % current system time
formations[] = getAllFormationFromDB();
users [] = getUsersWhichReplayedNegativ();
```

Activation Guards:

```
return users.isEmpty() && currentTime > = lastExecutionTime+604800;
```

Conditions:

```
return true;
```

Actions:

```
if (!activationGuards)
        foreach user in users[]
                proposeNewGroup(user.ID, user.Formation.Name );
        end foreach
end if
```

Rules Generation:

```
if (activationGuards)
        foreach formation in formation[]
                if(!checkDbIfGroupAlreadyExists(city))
                        createRuleR111(formation.getName())
                end if
        end foreach
        cloneRuleMTA321(currentTime)
else
        cloneRuleMTA321(lastExecutionTime)
end if
```

Rule ID:

R111

Rule Description:

Creates groups based on the study program of the users and inscribes students to the specific group where they belong to. Its structure is similar with the one of rule R101, containing just a slight change in the type of groups which are to be created.

Parameters:

```
programName     % contains the name of a specific study program, like "Master in
                  Computer Science"
```

Data Acquisition:

```
users[] = getUsersFromTheProgram (city);
```

Activation Guards:

```
return true;
```

Conditions:

```
return true;
```

Actions:

```
if (groupDoesNotExist(programName))
        createGroup (programName);
foreach user in users[]
        if (userIsNotPartOfGroup(groupID, user.ID))
                inscribeUserInTheGroup(groupID, user.ID);
        end if
end foreach
```

Rules Generation:

```
createRule201 (programName)
die();
```

Rule Description:

Adjusts groups that are either inactive or have lost a couple of member, and now they have less than 3 members, which is the minimum required to have a community. Resources of previous groups will not be available anymore as new ones will be created.

Rule ID:

MTA331

Parameters:

```
lastExecutionTime                       % unix timestamp time
```

Data Acquisition:

```
currentTime = getSystemTime();          % current system time
inactiveGroups [] = getInactiveGroups();
smallGroups [] = getGroupsWithLessThanThreeMembers();
```

Activation Guards:

```
currentTime > = lastExecutionTime+604800;
```

Conditions:

```
% the number of groups has to be odd in order to merge them
return ((inactiveGroups + smallGroups > 1) && ((inactiveGroups + smallGroups)%2 == 0))
```

Actions:

Rules Generation:

```
if (activationGuards)
        mergeGroups();
        cloneRuleMTA331(currentTime)
else
        cloneRuleMTA331(lastExecutionTime)
end if
```

The proactive cycle presented above is quite basic because it forms online communities of students which come from the same city. But what happens if there are members that **want to leave these groups**? Or about the case where users **want to join other communities** which seem more interesting?

Additional set of rules and scenarios have to be designed for addressing some special situations that may occur. On the actual Moodle system, lots of students have left an empty space where they should have indicated their city of residence. Maybe at the moment of completing their Moodle description they were not aware of the importance of this field. But after multiple notification messages received from the system, they decide they want be an active part in different communities and fill-out their current city. And finally, ***meta-scenarios MTA311*** and ***MTA111*** come into play – one checks once per week for users which have updated the city field and the other one inscribes them into the corresponding social group. They are strongly connected as MTA311 uses to activate MTA111. The activation process of MTA311 happens only once and afterwards it will continuously clone itself when needed.

Meta-scenario MTA321 uses to check once per week if there are students that do not want to be part anymore of certain groups. This rule is activated by one of the basic rules R203, which checks each time if MTA321 was already initiated. When the users are taken out of a social group they are asked by the system if they want to participate in social communities with other specific characteristics. One situation where these rules would apply is when a student does not want to stay inside the group of his/her city of residence because there is no activity inside this group or the number of members is too small.

Rule R111 is pretty similar to R101 because it has the simple task to create Moodle groups and get the students together in communities based on their study program. Multiple rules of type R111 will be instantiated as there are at least a couple of dozen different study programs at University of Luxembourg. Then the usual rules for creating specific resources would be activated by R111 and so on, until another cycle is finished.

A special role is taken over by ***meta-scenario MTA111*** because its purpose is to detect which groups are inactive or where there is only a small percent of their members that are involved in social activities. After detection, the inactive groups are either merged together or their users are encouraged to change the group's status by being more operative.

Figure 4.3 exemplifies a possible execution of the rules, which run in parallel with the basic ones, introduced at the beginning of this chapter. This is a great advantage because rules do not have to wait for other rules to finish their execution and so, actions like creating new groups can be done concomitant with removing members from other communities or merging smaller groups of people which are not so active.

In Annexes, A and B, a rule and a scenario are given as an example of who would they look when they are ready to be run by the proactive rules-engine. Java was chosen as the programming language for programming all the rules as well as the proactive engine. Their code respects the structure given above in pseudo-code except just a couple of methods that will change their names. These methods will still continue to accomplish the same tasks they were initially designed to do.

The benefices of creating such groups are better revealed with the help of two concrete situations involving students which are already inside the newly created communities. For the first case, an example where lots of students live in the same city and don't know about each other is taken. Assuming that the size of the city is quite small, the point of the example becomes more significant. Inside the group of people coming from this city students can share precious information related to transportation, good places where to

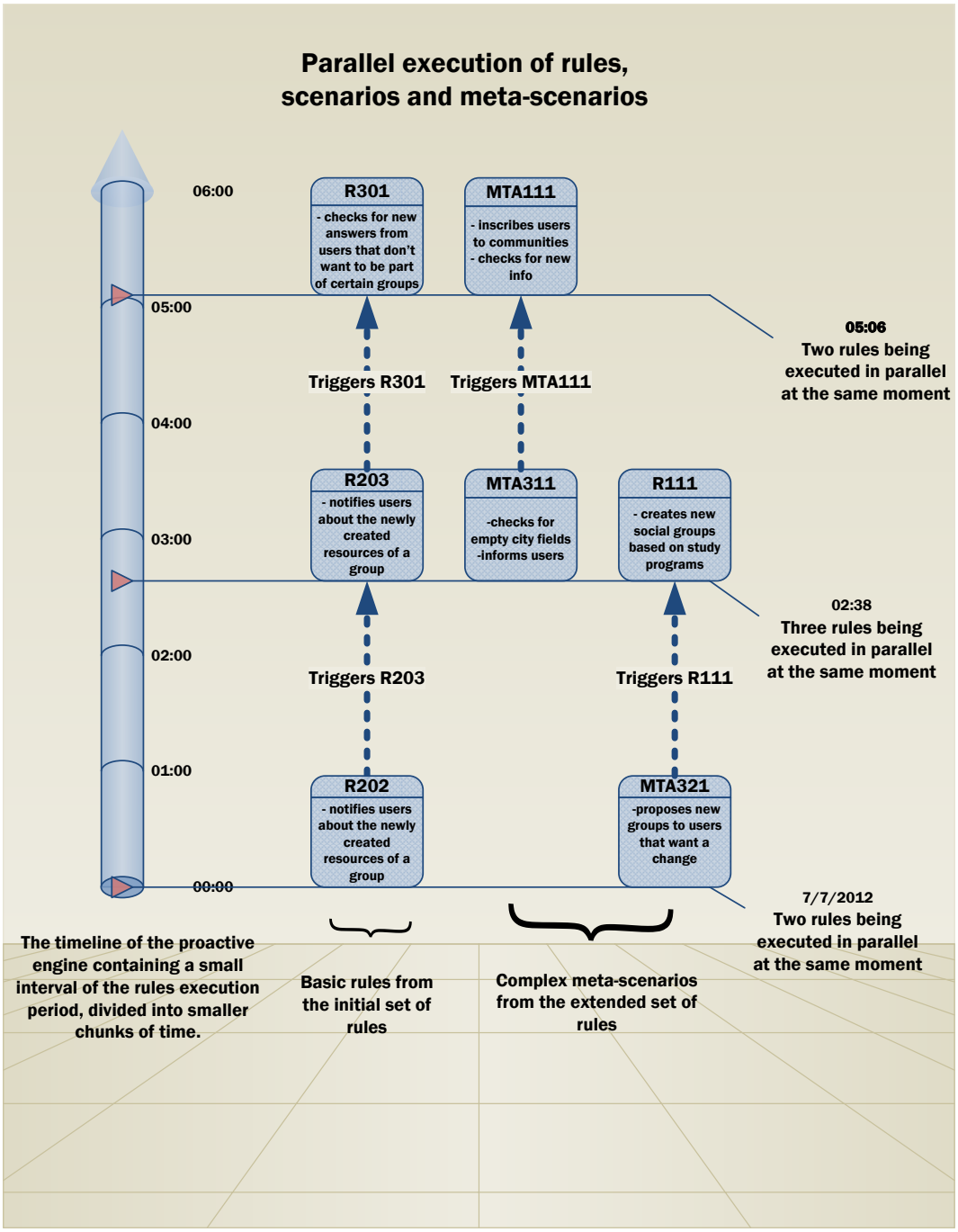

Figure 4.3: The parallel execution of different rules and meta-scenarios

eat, housing and other important domains. And speaking about transportation, maybe two students are neighbors and one comes by car at the university. They can arrange to go by car together and share the costs or agree to contact each other in case there are traffic jams and buses or trains cannot arrive to their destination without a big delay. This way, new opportunities are opened for the members of this community.

And for the second case, communities will contain students that are inscribed in the same study program. Inside these groups, they could share notes, assignments, compare results of various projects and get subjects from the previous partial and final exams.

It is quite difficult to find data common to the majority if users in the Moodle's

database that is sufficient for creating communities of students. It is crucial because it defines the goal of the groups and together with it all the necessary resources. The best source of common information for creating communities of students inside Moodle still remains the data collected from the user's profile.

There were so a couple of difficulties encountered when groups of people coming from the same city were created:

- Often, Moodle profile fields are not taken seriously by the students as they are not aware of its importance and so it is left empty

- And then, even if it is completed, there is no guideline on how to complete these fields or no list containing correct city names. So, multiple names were given for the same city or, for bigger cities, names of neighborhoods were written instead of the city name;

The solution chosen for solving these matters was to program a rule to be aware of all the special cases of city names and another set of rules to suggest users to fill out their Moodle description, while checking for updated city fields. It would be much easier if :

- Students would be made more aware of the importance of providing Moodle with correct information about them by providing a detailed guide with all the necessary steps required for having correct data on their profile;

- The process of completing the Moodle description would be mandatory.

4.4 Moodle

The first question to be asked is "Why use Moodle at all?" Classes have been running lots of centuries without using computer or the Web. Even though traditional, face-to-face lectures and courses can still be highly productive, supporting teaching with a wide range of online tools, opens up new possibilities for the learning process. Moodle is becoming very popular across the world thanks to its community and to the big number of known Moodle web sites, which is growing exponentially.

According to [J., 2005], published in 2005, more than 250 providers of commercial Learning Management Systems were identified on the market, plus around 40 other open source offerings. At that time the most known were Moodle, ILIAS, Eduplone, WebCT, Bscw, Claroline and SAKAI. Because of their wide developer communities, continuous support and improvements made to them, the majority of these systems are still on the market. They become potential competitors for the commercial products because of the growing interest in open-source platforms.

In a more recent study [McIntosh, 2012], released in 2012, differentiates between multiple groups of learning management systems such as Corporate LMS, Learning Content Management Systems, Educational LMS and others. Only the Educational LMS lists around 110 commercial products and almost 60 open source products. Probably the most popular in the open source category is Moodle, which is an abbreviation for Modular Object-Oriented Dynamic Learning Environment.

Open-source software is developing very fast and lots of non-profit institutions appear or are already at a mature phase of their development. Universities and other educational organizations are adopting open source platforms such as Moodle in order to manage their needs. This can be a very complex task because of the high level of complexity when using such a system. On the other hand Moodle is showing its complexity and offers on-line assistance for using the environment, providing examples for installing administrating and using properly this LMS.

Being an on-line course management system, Moodle provides diverse learning opportunities as well as a stable platform for distance learning courses. According to Wikipedia, distance education or distance learning is defined as field of education that is created especially for students that cannot be physically present in classrooms. Its main objective is to explore different teaching methods and technologies in order to develop the knowledge of students. A general definition is given for distance learning in [Honeyman M., 1993] – *"a process to create and provide access to learning when the source of information and the learners are separated by time and distance, or both"*. Moodle is now a stable LMS which has developed and improved over the past years. Besides the facts it was able to sustain itself and to continuously evolve through the Moodle society, it covers the majority of features and functionalities required from a LMS system. General reasons for picking Moodle as a LMS are expressed in the following subchapter 4.1.1 – "Pros and Cons of using Moodle" and then, a series of personal arguments for choosing Moodle are specified in subchapter 4.1.2 – "Reasons for choosing Moodle".

4.4.1 Pros and Cons of using Moodle

Because many things have been written about the advantages and disadvantages of Moodle, figure 4.4 is designated to highlight only a couple of the most important pros and cons of the open source/free learning system Moodle. Normally, pros and cons are important for people which have to pick a LMS, which sometimes is a very difficult decision. But for this work it is used to illustrate characteristics which are among the strengths of

the ideal system proposed in section 1 – "The Ideal System" of chapter 4 - Theoretical Background.

Figure 4.4: Pros and Cons for using Moodle

Identifying disadvantages of using Moodle is also very significant because it indicates exactly where improvements could be done to such a system. A good example would be the lack of community oriented groups in Moodle, which can be speculated and fixed with the help of a proactive system.

4.4.2 The reasons for choosing the Moodle platform

There were multiple reasons for choosing the Moodle environment as a test platform for proactive rules inside communities of practice. One of the most important reasons was that the proactive engine was already deployed aside the local Moodle system at the University of Luxembourg, where results were successfully collected each semester by professor Zampunieris and his team. The second reason was that student interaction can be monitored in a real learning environment where actions were not artificially triggered but were produced by individual users. An important criterion was that Moodle contains lots of the characteristics of an ideal system described in chapter 4.1 – "The Ideal System". It can be easily extended, plug-ins can be installed quite fast and results are collected in real-time. And finally but not the last reason was that Moodle dos not have a notification/ reporting system sufficiently developed to adapt the needs of the students, professors and administrators.

The main reason for selecting Moodle as the learning platform for developing our proactive meta-scenarios was the lack of social networks and means of interaction between the users, which can be regarded also as a drawback of using this system.

An initial decision towards using Moodle was taken by professor Zampunieris and his team because it was an e-learning web platform with a lot of useful and practical characteristics. More pros for choosing Moodle are listed above in chapter 4.3.1 – "Pros and Cons of using Moodle".

Currently, social interaction between users is very limited because Moodle provides only a few means of communication. Among these, there are possibilities of communicating inside courses via forums and chats, and outside courses via messages. All these forms are important because they can be used together to create a better social environment for the users.

Communicating efficiently is essential when speaking about on-line communities and about the learning process itself. Several means of communication are offered by the Moodle environment, where questions, ideas and thoughts take shape as a result of expressing the actual level of understanding of each course's subjects.

For forums, Moodle provides quite a variety of forum types [1]:

- A single, simple discussion – where only one discussion is opened in the forum. This choice keeps the dialog focused on one topic;
- A discussion for each person – each user is limited for starting only one conversation, which in return could have multiple replies;
- A Q&A forum – is a type of forum where users first have to post something to be able to see and reply to other postings;
- And the standard forum – where everyone can post and reply and were multiple discussions can take place;

Chats are tools that permit users to engage in an online real-time conversation. Specific functionalities are included for handling and reviewing dialogues. They can be added to specific courses in the weekly sections.

"People Block" is a used for displaying all the participants of a course. It is the fastest way of interacting with users that are enrolled in the same course. With a simple click on this block, a page is displayed where users can see more details about the participants of the same course.

[1] http://www.yorku.ca/laps/eso/facultystaff/pdf/ch6.pdf

And finally, the "Message Block" gives the participants of a course the opportunity to send and reply to messages, while being still online. This form of communication is not recommended inside courses but can be used while exploring other parts of Moodle.

The good news is that the existing set of communication tools and the visual interface can be easily extended with the help of Java Script and PHP. More details about these features can be found in chapter 5 – "Practical Implementation".

Because the main proactive engine was already set up and running, the main work of this thesis is to propose new scenarios and rules in the context of communities of practice in order to increase the level of interaction between users inside the Moodle platform.

4.4.3 Moodle Groups

Moodle offers two main ways for getting users together for common activities or tasks - groups and groupings. Technical details about these Moodle features are skipped because they offer no particular interest. More important is the reason why were they chosen for this project and how can they be used for applying the theoretical concepts.

Creating a new form of reuniting users is a bit trickier because it has to be well integrated with all the other Moodle features like roles, visibility, etc., and in the same time not to interfere with the other plug-ins and running processes. As the purpose of our proactive engine is to leave the initial software system unmodified, a decision was taken for using the existing features of Moodle.

Groups are Moodle components which allow a lecturer or a teacher to create one or more groups and inscribe users into these groups. The users are united in achieving a common goal that is expressed under the form of an activity, a subject or task, by accessing specific resources. Groupings are also Moodle components that allow Groups of users to be distinguished and separately organized. What is then the difference between a Group and a Grouping? Making a long story short, groupings are collections of groups. The idea is that users are allocated to Moodle Groups and the Moodle Groups are allocated to Groupings.

Groups and groupings are right tools for applying the concept of communities of practice in an on-line e-learning environment. The structure of Moodle Groups and Groupings, plus the purpose for which they are created, represent a very close applied form of communities of practice.

The concrete idea of this Master Thesis was to create a course inside the local Moodle platform only for social purposes. Then students with an active account would be organized into Groups created for different purposes. For example, a group would be created for people inscribed in the same study program at the University level. Then all sort or resources like forums, chats and documents would be made available to them. But here comes the problem - how to make specific resources available only to a certain category of people? The answer, in this case, was quite simple: the only way of making activities accessible to a set of students is to create Groups and Groupings.

Teachers or course managers can decide if a course is using groups or not, and, more important, if these groups are separate or visible. The difference is that in separate groups users can only see participants of their own groups, meanwhile, in visible groups users are working in their own groups but they have access to other groups.

In cases when maximum privacy is needed, separate courses need to be created in order to avoid any information leaks. This case is not possible for our scenarios because the number of courses needed for creating social activity would increase exponentially. Already it is very difficult to manage resources for each community inside Moodle.

Let's take a very simple and concrete example, which is a part of the solution proposed

in chapter 4.2: a different group is automatically created for students coming from the same city, if there are more than 3 students with the same city name written in their Moodle description. The University of Luxembourg has students from 100 different countries [2] and these increases the change of having a big number of groups. For each group, specific resources will be created. According to simple math calculation, if for each group minimum 3 resources are created (a forum, a chat and a Q&A questionnaire) and more than 50 groups to be created, a minimum number of 150 resource components will have to be created. And this entire process of creating resources would take part while the Moodle system is running. The use of these resources and the intense participation inside the groups will push the system and the database to their limits.

All these examples and features will be simulated and integrated on the local Moodle platform at the University of Luxembourg. More about these future plans in subchapter 7.3 – "Future Research".

[2] http://wwwen.uni.lu/university/about_the_university/5_good_reasons

Chapter 5

Practical Implementation

The practical implementation phase, currently under "construction", follows already the guidelines defined by Professor Zampunieris and his team because a considerable number of scenarios and meta-scenarios were already deployed on the rules-running system which worked together with the University of Luxembourg's local learning platform.

It is well known that Moodle is a VLE based on the PHP language. The code is not seen by the browser because the majority of the operations are happening on the server side were the PHP code is run. All the content is send to the Client, which is nothing but a simple web page. Moodle supports request/response models, so, in some special situations, the Client sends both synchronous and asynchronous request to the Server with the help of AJAX. The response normally comes as an entire HTML page from the server.

The database plays a key role in any LMS, as it stores all the data used related to users and their actions. But not all they key information is stored in the initial tables so new tables have to be designed in a new database to house all the extra data.

Figure 5.1 contains the most relevant schema which shows how the proactive engine is currently integrated with Moodle at the University of Luxembourg for the assignments mechanism. The red parts represent the elements which are not part of the initial architecture. On the client side, the extensions include a Moodle block, where a short list of groups will be shown and special page to express the content of each particular community.

In order to make Moodle a proactive system and to develop internal CoPs, modifications need to be made both on the client side and on the server side. Each one is extended according to a predefined plan because all the developed parts need to be compatible and to work together for having a proper output.

Moodle uses a client/server architecture which consists of several elements: the Client with a GUI developed in PHP and Java Script, the Server – Moodle, and a unique local database. Extending the client side involves programming done in PHP in order to extend the graphical user interface. As for extending the server side involves writing in Java the proactive rules, i.e. given in pseudo-code in chapter 4 - Theoretical Background, creating new tables in the proactive database and creating PHP scripts for internal Moodle tasks (see PHP Notifications Script).

The architecture of the entire system is quite intuitive and it is designed in such a way that it can work with many other software systems, especially the ones which have a client/server architecture.

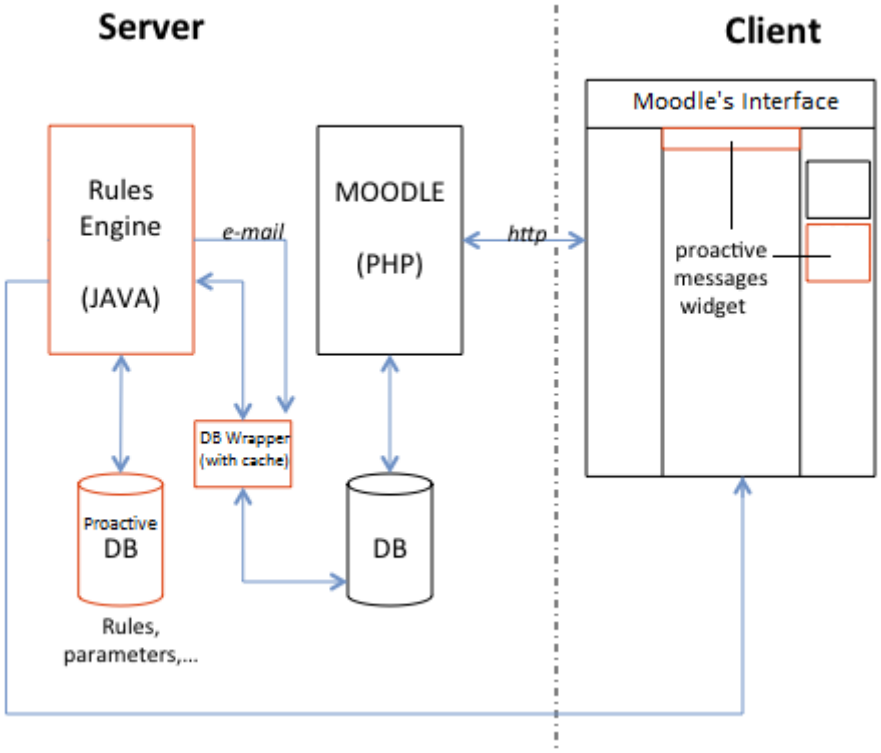

Figure 5.1: System Architecture

5.0.4 Modifications on Client/Server side

On the Client side there are quite some modifications left to be implemented. Even though grouping students is already handled by the server side via Moodle Groups/-Groupings and the whole structure of a community inside Moodle is looking like a normal course, additional elements of the GUI need to be implemented. Inside their group, each student should have access to:

- A Moodle side block containing all his/her social communities (e.g. see figure 5.2);
- The complete list of all the members of a specific group;
- Special tools for asynchronous and synchronous communication like text chats, forums and instant messaging;
- Collections of Wikis (i.e. a web page whose content can be edited) where students can make public modifications to existing topics;
- Shared data, such as relevant documents, common projects or general discussions;

From the list above, the "Group Block" needs a bit more attention. It will be similar to the "People Block" which already exists in Moodle. Its purpose is to display a small list containing only the name, a graphical representation and a "remove" button. The "remove" button removes a student from a social group but only after the user give his/her permission the second time. Moodle blocks are displayed either on the left side or on the right side of web page. A user needs to click on a single group from the list of social groups to display all its content on a new page. Preferably, a visual alert will be displayed next to each group to indicate if there are new activities in the group or new

tasks have been assigned. The visual alert will be collared according to the state of each group and it will contain distinctive and obvious signs for users to understand if one of their communities is active or not.

Figure 5.2: Groups Block inside Moodle's interface

A main page of a group is actually the main page of the course, which hosts all the social communities, with the small difference of displaying only resources which are related to that group.

On the server side, the only real modification left to be done is the creation of a GUI for the rules engine. The other steps of creating social groups and registering this information in the database are the job of the proactive rules which have to be written in Java and the job of PHP scripts that are executing actions inside the Moodle environment.

All the rules presented in this thesis will run on top of the rules engine, which is currently implemented using the Java language at the University of Luxembourg. The dynamic rules running system (RRS) was proposed by professor Zampunieris in his paper [Alami et al., 2008], which is introduced in chapter 3 - Review of the State of the Art.

A rule may have important tasks such as calling a PHP script on the server side and executing very complex queries to the database or they can be quite simple, sometimes with only one action to perform.

The practical implementation of the rule R101 that was in charge for creating social groups based on the city of origin of students is given in PHP Notifications Script. It extends the AbstractRule.java, which contains five abstract methods that are declared but not implemented: dataAcquisition(), activationGuards(), conditions(), actions() and rulesGeneration().

In figure 5.1, there is still place for one important component – the graphical user interface for the proactive engine. Building this interface is mentioned in Conclusions as future work because it is not a vital part of the whole system.

Chapter 6

Conclusions

6.1 Final Conclusions

This thesis is a follow-up of the efforts made by professor Zampunieris and his team at the University of Luxembourg. They started deploying and using the proactive engine aside the local Moodle platform since 2006. And from then on, encouraging results have been obtained to prove that proactivity represents a valid solution for improving the learning process of a high percentage of students.

The main objectives of this thesis were: to introduce concepts like CoPs, proactivity, situation and context awareness, while highlighting the major advantages of these topics, to propose solutions for enhancing social life inside Moodle, analyze the reasons of choosing a LMS for proving the most important ideas and to establish the importance of bringing people together in a online environment as well as making a system proactive and aware to special events and situations.

Initially, the work of professor Zampunieris and his team was focused more on the assignment system on Moodle. The deployment of the rules-engine did not affect other operations running on the Moodle system because rules were primarily used for helping and guiding students in their continuous learning process. Introducing CoPs to Moodle is a bigger and a more complex task, as the expected outcome is different. The proactive cycle, composed of various sets of rules, will modify the status of the system meaning that for the first time the proactive system will change structures inside the learning platform, add new resources to it, extend the GUI and insert by its own new data into Moodle's database. OCoPs are much stronger and more useful when proactivity and situation-awareness are used together for common tasks.

The final purpose of this work is to improve the social life of students inside Moodle by creating new ways of bringing together groups of students. Important steps towards creating social activity inside Moodle have been taken with the creation of Groups and Groupings in the newer versions of this LMS but they are designated to work only at the level of courses.

Moodle itself is an environment which proves the usefulness of OCoPs. It primary goal is to focus on improving the learning process of students and their knowledge. But it is currently limited because it misses the existence of more advanced social groups, where other kind of interaction is developed.

Proposing initial reasons for organizing groups of people online was not easy because the purpose was not only to improve their learning skills but to develop a certain level of social interaction between them. Until now, the most relevant ways of getting students together in on-line communities on Moodle were to group them by the city of residence

and by their study program. And issues did not stop at the stage of finding good reasons to bring students together but continued when rules had to be proposed as a solution for creating social cooperation inside Moodle. Complex database queries were introduced to solve the case when city fields in the description of a user are left empty. Also complicated analyses are conducted inside the body of rules for special cases when, instead of correct names for cities, abbreviations or just cities names which do not exist are found in the database. Learning how to avoid and fix certain situations helped a lot in understanding better how the whole e-Learning system works and what are its strong and weak points.

At the beginning of this project, two ways of getting users to participate in social groups were discussed. One method assumed asking each student in particular if they would like to join the initiative of creating communities while the other method proposed a more optimistic approach. Instead of asking each user if he/she wants to be part of the initial circle of students, the proactive system will enroll users directly into their corresponding communities. The main idea behind this reason was to directly engage students in mutual social interaction as they are encouraged to discover how a community looks like and how to use local resources like forums, chats and available documents. There is a differentiation between these methods only for creating initial circles because it represents the start point of the whole cycle. Afterwards, users will be allowed to request themselves to enter in different groups or to leave them. So, these methods will finally mix and the proactive engine will use both of them for initiating other forms of bringing users together.

Students represent a strongly connected community. Sometimes, interaction between them is very difficult as they do not have enough time to know each other personally. At the University of Luxembourg there are students which come from all around the world. Each student has his/her own account which is created with real data provided when they enroll for the first time in a study program. Using unique accounts with credible information for each student offers a kind of assurance and increases the level of trust of all participants.

Transforming Moodle's behavior into a proactive behavior is very important because:

- Moodle is a complex platform to work with and it was not possible to anticipate all required routines and behaviors. Often, this task was left to Moodle's administrator and to the teachers which had to use their own initiative for taking decisions;
- If the system is waiting for explicit commands it will become inefficient and ineffective;
- Pushing users to adapt to new situations, will make them more aware and will make them take more responsibility for enhancing the learning environment;
- Helping, guiding and assisting students offers Moodle a great advantage over other learning management system;

For all the reasons mentioned above, the proactive behavior will not only improve the learning platform but will serve much better the user in his/her daily activities.

A vast amount of information and data about student communities and their collaboration inside these circles will be accumulated over a large period of time. Here is where the proactivity will come at hand by making use of all the gathered data. First of all, relevant data will be used to determine exactly how many students got really involved inside Moodle's social groups, if some documents or resources were intensively used and others were not, and, finally, to analyze if the answers from surveys and on-line questionnaires are enough for creating strong conclusions.

A very difficult task was to choose which concepts to explain from many related research fields. These explanations are given to indicate the direction of this research and to provide a small introduction to various topics. The background literature is so big, that only the key concepts were discussed and analyzed. Hopefully, they are enough for the reader to understand were each idea plays an important role for finding a viable solution to apply OCoPs, proactivity and context awareness inside an LMS.

Even though Moodle is far from being the ideal system for deploying our proactive system, it has a lot of useful functionalities which will be used for achieving the desired output. Features like Groups and Groupings will help organizing students into circles, while tools like forums, chats and messages will prove their usefulness inside the learning platform. But maybe the best thing about Moodle is that it can be easily extended with the help of plug-ins, if the environment does not have already integrated the necessary tools or it misses some important features.

Universities from all around the world are confronted with problems when they have to provide services to students like accommodation, transportation, jobs, places where to eat, cultural events, sports activities and the list does not stop here.

If the example of student housing is taken, which is an essential requirement for a student's ability to focus, concentrate and study well, multiple challenges have to be considered before a final decision is taken for paying for an accommodation. These challenges come as a consequence of the lack of rooms that are situated nearby the learning institution, of high prices, of a high demand for student housing,

In the majority of cases, senior students have solutions to these problems because they were confronted with them since the beginning of their studies. Newcomers can benefit from the knowledge of other students. These details are often omitted by the staff of the universities as they do not take advantage of existing knowledge. Nowadays, despite the vast amount of on-line interaction between students, the work which is done outside the established boundaries of a learning platform is quite low. By creating on-line communities many of the students issues will be addressed. Both the educational institutions and their members will benefit from this solution. It will help the users in achieving their goals, regardless if they want to improve their learning process, solve common problems or just meet new interesting people with which they have something in common. On the other hand, universities will improve their student services without having to pay and train extra staff.

Having an on-line environment, where knowledge is shared, discussed and rated between users, represents a big advantage for corporations, organizations and public institutions.

6.2 Future Work

This section contains logical steps which will be taken after this Master Thesis is finished, both at a smaller and larger scale. Future work will include the discovery of new patterns for organizing CoPs, a better and efficient structure for proactive rules, a more sophisticated visual support inside Moodle, a GUI for the proactive engine and finally, the collection and analysis of relevant results from the proactive engine. A close collaboration will be maintained with professor Zampunieris and his team at the University of Luxembourg in order to continue the ideas of this work.

Finding new patterns to apply proactive rules

Initially, the most edifying patterns discovered for forming communities of students were based on the city of residence of the students and the different study programs in which they were inscribed. These motifs were strongly related to the information which could be extracted from Moodle profiles. Only 2 patterns are not enough to generate intense social interaction between users. So, they have to be extended and new solutions for building groups of students need to be found. Finally, it is resumed at finding relevant data that binds users and which can be applied to more than just a couple of users.

One initial idea was to assemble those students which have common interest points inside the on-line learning platform. For example, students that read articles from different domains will be able to discuss and to share their point of view – maybe this will turn out to be a great review for another user. Another idea would be to organize communities based on the sports activities provided by the University of Luxembourg. At the moment, a separate on-line form is available to persons learning or working at the University of Luxembourg for inscribing in various sports activities. It would be much easier to integrate these forms in the local Moodle platform both for students and for the qualified staff.

Of course, new patterns have to be relevant to the context they are applied to and anticipate the reasons for which students will want to take part in these communities. Having a big number of social groups for each user will decrease the importance of having communities because the idea is to have students spending a small amount of time inside on-line communities as their primary goal in a LMS still remains to learn and finish tasks given by their professors.

Designing better and more efficient rules

The rules created for the initial proactive cycle are just the start of using CoPs in LMS. They are necessary for developing and maintaining basic communities inside Moodle. More advanced set of rules are to be proposed in the near future that will complete the current structure for creating social groups.

Improvements have been done at the level of a rule's phase of acquiring its parameters by a lazy evaluation process as described in section 2 - Proactive Learning Management Systems of chapter 3 - Review of the State of the Art. But nevertheless, maybe there are ways to improve the structure of a rule by merging some of its parts or just by reducing their execution steps. A better grouping method will allow future improvements at a structural level. For the moment the structure of a rule is composed of five parts: data acquisition, activation guards, conditions, actions and rules generation. Already, many types of rules exist because they serve a lot of different purposes, but they still rely on the same structure. Frequently, some of the parts of the rule serve no purpose as they only perform simple actions. So, when the system runs a big number of rules in parallel, having a small execution time for each rule counts a lot in the economy of the whole system.

Improving the visual support inside Moodle Groups

Having a good visual support is a crucial characteristic of any system. The actual layout influences students in their decision of using more often or not the online environment. At the moment, Moodle's GUI can be quite charged with information that users do not really need. This information should only be displayed on user's explicit command or when it becomes vital for a student (e.g. when a student has to inscribe to final exams

a strongly visible warning should be displayed on every page of his/her account). In general, it is hard to tell which information is better to be displayed on a student's account, so this decision should also be influenced by the user (i.e. the GUI should be highly customizable).

The final outputs of proactive rules are shown through Moodle's graphical-user interface. Either messages from the system, newly created documents or fresh assignments need to be seen by each user in order to trigger an action from his/her side. Moodle's GUI represents the client of the entire architecture and it is strongly connected to the server side, from where it receives all its data. In this particular case, the interface of Moodle will be changed to better address the needs of its users.

Collecting results

After deploying the scenarios on the local Moodle system at the University of Luxembourg, a set of results will be gathered for the whole winter semester. Users, which will be represented by real students, will contribute to the data collection by interacting with the Moodle environment. Moodle is a real-time running system where many students make different actions in the same time, so, all activities will be tracked and examined by the proactive system.

These results will be analyzed and then published for the whole research community for getting reviews and making them more aware of the work done in the fields of proactivity and CoPs. After collecting initial results, overall knowledge will improve and will lead to other ideas for creating new groups of students, addressing better to their needs.

Results speak for themselves in many research cases and they prove if the experiments that were conducted are successful or not. Their interpretation covers all collected data and will reveal if building student communities was done the right way.

Using experimentally obtained results will offer a big advantage because real data will be analyzed and evaluated. Precise evaluations will establish the actual consequences of using CoP, proactivity and situation-awareness together in a LMS. This procedure is not possible in the case of theoretical methods, where only probabilities and predictions can be used to estimate the future impact and the usefulness of the applied solutions.

Applying the proactive CoPs in other environments

The Moodle platform is just the start of applying communities of practice in learning management systems, and why not in other software platforms? Companies and organizations should benefit from software which will allow all their members to network, share knowledge and identify mutual solutions.

It is very important to explore the potential of extending social processes of existing software platform to a wider range of collaborative activities. Long processes like learning and key assets like knowledge represent the core of strategic thinking and applied success in multiple communities of people. Knowing how to embed knowledge into practice offers a great advantage for an organization as it can generate value for its line of work.

Hopefully, more institutions will realize the importance of having proactive software platform with well-defined hierarchies of users inside their networks of people.

Creating a GUI for the RRS

The proactive engine is designed to run and manage itself without any external help. But, for example, tasks like detecting its compatibility with other software or creating a report of all the actions which were performed inside the system take a lot of time. Their

execution can slow down the proactive as they require a lot of resources to be performed, so their execution should be controlled and somehow limited. Through a powerful GUI, the proactive engine could be handled and supervised by a trained person. The benefits of such a visual component are well-known. To start with, the initial interface will be simple, allowing people to check log files or see which rules are currently running. After having this basic visual support, a more complex interface will be designed. It will allow the administrator to directly interfere in exceptional cases in the matters of the proactive rule-engine. The system cannot take decisions from situations that it is not aware of, but of which a system manager would know by other means. For example, during execution, some of the system's resources would be used by multiple rules which will create a conflict. It will be immediately notice by the engine and reported through the graphical-user interface from where the administrator will decide either to stop some rules or to give them more priority.

As the rules-engine manages, monitors and controls all the rules being deployed, any inconsistency should be detected immediately due to the situation-awareness property. Regardless of how the system will decide to solve its problems, the administrator would have the last word to say about these special situations. Reporting and signaling inconsistencies like a longer execution of a rule are vital when a big number of rules are running at the same moment.

Appendix A

Rules Examples : The Abstract Rule

```
package rules;

import ruleRunningSystem.QueueManager;

/**
 * <b>(Abstract) Definition of a rule, rules are a specification of this
     class.</b><br>
 * based on the RuleTemplate class defined for version:
 * <ol><li>renamed
 * <li>extended</ol><br>
 *
 * @author Sandro Reis
 * @version 3.0 - Sandro Reis 2011
 * @version 2.0 - Sergio Marques Dias <br>
 * @version 1.0 - Yann Milin
 *
 */
public abstract class AbstractRule {
        private long id;
        protected boolean activated;
        protected QueueManager engine;

        // default value, to be changed on the constructor if needed
        protected RuleType type = RuleType.SCENARIO;

        /**
         * default constructor
         */
        public AbstractRule() {
        }

        /* ******************************************************
         * ******************************************************
         * ******************************************************
         * Abstract methods come here
         * ******************************************************
         * ******************************************************
         * ******************************************************
         */

        /**
```

```
     * Get information from the LMS (Moodle DB) in order to use these data
         in its other parts
     */
    protected abstract void dataAcquisition();

    /**
     * Performed after the data acquisition part and is made of a set of
     * AND-connected tests on local variables that, once evaluated,
         determines
     * if the conditions and actions parts will be performed afterwards
     * <br>
     * If all the activation guards are evaluated positively, then the
     * conditions and actions parts are performed.
     * <br>
     * Boolean variable called "activated" set accordingly to the result
     * of the activation guards evaluation
     *
     * @return result of the evaluation
     */
    protected abstract boolean activationGuards();

    /**
     * Made of a set of AND-connected tests on local variables that, once
     * evaluated, determines if the actions part will be performed
         afterwards.
     * <br>
     * Conditions tests syntax and semantics are equivalent to activation
     * guards tests.
     * @return result of the condition's evaluation
     */
    protected abstract boolean conditions();

    /**
     * Made of a list of instructions that will be performed in sequence if
     * all the test of the conditions part are evaluated positively.
     */
    protected abstract void actions();

    /**
     * Performed at the end. It allows the rule to generate other(s) rule(s)
     * that will be performed afterwards.
     */
    protected abstract boolean rulesGeneration();

    /**
     * overrides Object.toString()
     * implementations of AbstractRule are obliged to implement a specific
         toString Method!
     *
     */
    @Override
    public abstract String toString();

    /* ******************************************************
     * ******************************************************
     * ******************************************************
```

```
 * FINAL methods come here
 * ********************************************************
 * ********************************************************
 * ********************************************************
 */

/**
 * @param rule the instance of the rule to add to the queue to run in
    the next iteration
 */
public final void createRule(final AbstractRule rule) {
       // if parameter is empty, do nothing
       if (rule == null)
              return;

       // set the new rule's engine to this, if not yet set
       // so that, afterwards, in the new rules execution, this.engine
           is not null
       if (rule.engine == null)
              rule.setEngine(this.engine);

       // delegate on the right engine method to add the rule in the
           right place
       if (this.engine != null)
              this.engine.addRule2Queue(rule, this.engine.nextQueue);
       else
              System.out.println("**[AbstractRule] createRule(): engine
                  is null. Not able to add rule '" + rule + "' to
                  rule's engine queue");
}

/**
 * @return RulesGeneration return value
 */
public final boolean execute() {
       String str = "**[AbstractRule] (" + this.getType() + ") ";

       dataAcquisition();
       if (activationGuards()) {
              this.activated = true;
              str += "activated. ";
              if (conditions())
                     actions();
       }
       else
              str += "not activated. ";

       str += this.toString();
       System.out.println(str);
       return rulesGeneration();
       // END of Simplification FINAL
}

/**
 * @return activated
 */
```

```
public final boolean getActivated() {
	return this.activated;
}

/**
 * @return Id
 */
public final long getId() {
	return this.id;
}

/**
 * @return the ruleType
 */
public final RuleType getType() {
	return this.type;
}

/**
 * @param act to set activated
 */
public final void setActivated(boolean act) {
	this.activated = act;
}

/**
 * @param engine the engine to set
 */
public final void setEngine(QueueManager engine) {
	this.engine = engine;
}

/**
 * @param i to set the Id
 */
public final void setId(long i) {
	this.id = i;
}

/**
 * @param type the ruleType to set
 */
public final void setType(RuleType type) {
	this.type = type;
}

/**
 * if deadline is not set<br>
 *   assignment is valid if we're "inside" a semester (start of course
    + one Semester)<br>
 * else<br>
 *   assignment is valid if deadline is not yet reached<br>
 * @return true if assignment is still valid in terms of dates
 * @see Wait#oneSemesterInSecs
 */
```

```
        protected final boolean isAssignmentValid(long assignment_id, long
            course_id) {
                long deadline = engine.db.getAssignmentDeadline(assignment_id);
                long now = System.currentTimeMillis()/1000;

                if (deadline == 0) {
                        long startdate = engine.db.getCourseStartDate(course_id);
                        return (now < (startdate +
                            lu.uni.fstc.utils.Wait.oneSemesterInSecs));
                }
                else
                        return (now < deadline);
        }
}
```

Appendix B

Rules Examples : R101

```
package rules;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.Date;

/**
 * @author remus.dobrican
 *
 */
public class R101 extends AbstractRule {

	private String groupName;
	private ResultSet students;

	/**
	 * Default constructor, mandatory for Hibernate to build object
	 */
	private R101() {
		setType(RuleType.RULE);
	}

	/**
	 * @param assignment_id
	 */
	public R101(string cityName) {
		this();
		setGroupName(cityName);
	}

	/**
	 * @see rules.AbstractRule#dataAcquisition()
	 */
	@Override
	protected void dataAcquisition() {
		this.students = engine.db.getStudentsFromCity(groupName);
	}

	/**
	 * @see rules.AbstractRule#activationGuards()
```

```
 */
@Override
protected boolean activationGuards() {
	return engine.db.groupExists(this.groupName);
}

/**
 * @see rules.AbstractRule#conditions()
 */
@Override
protected boolean conditions() {
	return true;
}

/**
 * @see rules.AbstractRule#actions()
 */
@Override
protected void actions() {
	if (!activated){
		java.util.Date date= new java.util.Date();
		SocialGroup grp = new Group (groupID, new
		    Timestamp(date.getTime()));
		engine.db.createGroup(grp);
	}
	while (!this.students.isClosed() && this.students.next()) {
		if (!engine.db.userIsPartOfGroup(this.groupName,
		    this.students.getLong("id"))){
			engine.db.inscribeUserToGroup(this.groupName,
			    this.students.getLong("id"));
			System.out.println("[R101] inscribing student ["
			    + id.toString() + "] to group [" +
			    this.groupName + "] ");
		}
	}
}

/**
 * @see rules.AbstractRule#rulesGeneration()
 */
@Override
protected boolean rulesGeneration() {
	try {
		if(activated){
			createRule(new R201(this.groupName));
			System.out.println("[R101] creating necessary
			    resources for group [" + this.groupName + "]
			    ");
		}
	} catch (SQLException e) {
		e.printStackTrace();
		return false;
	}
	return true;
}
```

```
	/**
	 * @see rules.AbstractRule#toString()
	 */
	@Override
	public String toString() {
		return "R101 - New group created. GroupName = " + this.groupName;
	}

	/**
	 * @param setting the GroupName
	 */
	public void setGroupName(long assignment_id) {
		this.groupName = groupName;
	}

	/**
	 * @return getting the groupName(
	 */
	public long getGroupName() {
		return this.groupName;
	}
}
```

Appendix C

PHP Notifications Script

Listing C.1: Example PHP Code

```
<script type="text/javascript"
    src="http://code.jquery.com/jquery-latest.js"></script>
<script type="text/javascript" content="charset=iso-8859-1">
        var c=0;
        var lastMsgId = 0;
        var paramtopass1;
        var newMessage=false;
        var t;
        var tMessage;

        //changes message from unread state to read state
        function readMessage(event){
                var wholeId = event.target.parentNode.id;
                var temp = new Array();
                temp = wholeId.split('___');
                var idMsg      = temp[1];

                //if it's first time a message is read
                //if(document.getElementById(wholeId).style.backgroundColor!=''){
                document.getElementById(wholeId).style.backgroundColor = '';
                document.getElementById(wholeId).style.fontWeight = '';

                $.ajax({
                        type: "POST",
                        url:"/blocks/coachingmessages/" +
                            "scripts/PamMessageIsRead.php",
                        data: 'msgId='+idMsg,
                });
                //}
                alert(temp[0]);
        }

        function deleteMessage(event){
                clearInterval(t);
                var wholeId = event.target.parentNode.id;
                var temp = new Array();
                temp = wholeId.split('___');
                var idMsg = temp[1];
                $(document.getElementById(wholeId)).remove();
```

```
        //appel du script pour effacer le message
        $.ajax({
                type: "POST",
                async: true,
                url:"/blocks/coachingmessages/" +
                    "scripts/PamDeleteMessage.php",
                data: 'msgId='+idMsg
        });

        lastMsgId = 0;
        var table = document.getElementById('commentaire');
        var rowCount = table.rows.length;
        if (rowCount==0) newMessage=true;
        else newMessage=false;

        //timedCount();
        t=setTimeout("timedCount()",1000);
}

function timedCount(){

        //since query searches by last__msg_id, first time is zero so
            loads everything afterwards just the new ones
        paramtopass1 = {"last_msg_id" : lastMsgId};

        $.getJSON(
        '/blocks/coachingmessages/scripts/PamGetMessages.php',
        paramtopass1, function(data) {

        //empty table if it was a delete to refresh
        if(!newMessage){$(commentaire).empty();}

        $.each(data, function(index, array) {
                tempBody=$('<div />').html(array.body).text();
                tempBo=tempBody.replace(/"/g, "'");
                var tempId = tempBo+'___'+array.msg_id;

                $('#commentaire').prepend('<tr id="'+tempId+
                '" style="cursor:pointer;"><td style=" width:160px;
                font:arial; font-size:10px; border-bottom-style: solid;
                    border-bottom-width:1px; border-color:c8c9c7;"
                    onclick="readMessage(event)">'
                +array.subject+'</td><td title="delete"
                    style="font-weight:bold;"
                    onclick="deleteMessage(event)">x</td></tr>');

                //add a "!" if important message
                if(array.importance=="1"){
                        $("#commentaire tr:first").prepend('<td
                            title="important" width="1px"
                            style="font-size:18px; color:#cf0000;
                            font-weight:bold;">!</td>');
                }
                else{
```

```
                $("#commentaire tr:first").prepend('<td
                    width="1px"></td>');
            }

            //changes color of background if message never read
            if(array.isRead==0){
            document.getElementById(tempId).style.backgroundColor =
                '#d5f9cd';
            document.getElementById(tempId).style.fontWeight = 'bold';
            }

            //new message loaded for notification bar purposes
            if(newMessage){
                $('#message').fadeIn('slow');
                setTimeout("$('#message').fadeOut('slow');",
                    5000); //pause for x seconds and then fade out
            }
            lastMsgId=array.msg_id;
        });

        newMessage=true;
        });
        t=setTimeout("timedCount()",1000);
}

//put newMessage to true if by starting it's empty so afterwards if we
    get a message popup shows
function testTable(){
        var table = document.getElementById('commentaire');
var rowCount = table.rows.length;
        if (rowCount==0) newMessage=true;
        //else newMessage=false;
}

$(document).ready(function() {
        newMessage=false;
        timedCount();
        tMessage=setTimeout("testTable()",1500);
        window.onload = confirmUserConnection;
        window.onbeforeunload = confirmExit;
});

function confirmUserConnection(){
        $.ajax({
                type: 'POST',
                url: '/blocks/coachingmessages/' +
                    'scripts/PamUserConnected.php',
                async: true
        });
}

function confirmExit()
{
        $.ajax({
                type: 'POST',
```

```
                    url: '/blocks/coachingmessages/' +
                        'scripts/PamUserOffline.php',
                    async: false
                });
}
</script>
```

Bibliography

Lithium Technologies Inc. 2012. Online support communities: Best practices and deployment tips for reducing costs and increasing overall customer satisfaction, 2012. URL http://www.lithium.com/pdfs/whitepapers/Online-Support-Communities_GAW_WD9pRA2z.pdf. [Online; accessed 15-July-2012].

Marc El Alami, Nicolas Casel, and Denis Zampunieris. An architecture for e-learning system with computational intelligence. *The Electronic Libray Journal*, 26(Artificial intellige):318–328, 2008.

John S. Brown and Paul Duguid. Balancing Act: How to Capture Knowledge without Killing It. *Harvard Business Review*, 78(3):73–80, 2000.

Sergio Coronado and Denis Zampunieris. Towards a proactive learning management system using early activity detection. In Array, editor, *SITE08 - Society for Information Technology & Teacher Education International Conference*, volume 1, pages 306–311, Las Vegas, USA, 2008. AACE.

Sergio Coronado and Denis Zampunieris. Continuous proactivity in learning management systems. In *EDUCON 2010 - The Future of Global Learning in Engineering Education*, volume 1, pages 201–205, Madrid, Spain, 2010. IEEE.

J. Michael Crant. Proactive Behavior in Organizations. *Journal of Management*, 26(3): 435–462, June 2000. doi: 10.1177/014920630002600304. URL http://dx.doi.org/10.1177/014920630002600304.

M.D. Eric S. Toner. Creating Situational Awareness: A Systems Approach, 2012. URL http://www.ncbi.nlm.nih.gov/books/NBK32848/. [Online; accessed 10-August-2012].

Pat Gannon-Leary and Elsa Fontainha. Communities of practice and virtual learning communities : benefits, barriers and success factors. *Elearning Papers*, 2007. URL http://nrl.northumbria.ac.uk/2147/.

A. M. Grant. Relational job design and the motivation to make a prosocial difference. *Academy of Management Review*, (2):393–417, 2007. URL http://intrinsicmotivation.net/SDT/documents/2007_Grant_AMR.pdf.

A. M. Grant and S. J. Ashford. The dynamics of proactivity at work. *Research in Organizational Behavior, 28*, pages 3–34, 2008. URL http://ec.europa.eu/education/archive/elearning/doc/studies/market_study_en.pdf.

Miller G. Honeyman M. Agriculture distance education: A valid alternative for higher education? *Proceedings of the 20th Annual National Agricultural Education Research Meeting*, December 1993.

Massy J. Study of the e-learning suppliers' "market" in europe. *DG Education and Culture*, 2005. URL http://ec.europa.eu/education/archive/elearning/doc/studies/market_study_en.pdf.

Gabriel Jakobson, John Buford, and Lundy Lewis. Situation Management: Basic Concepts and Approaches. In William Cartwright, Georg Gartner, Liqiu Meng, Michael .. Peterson, Vasily .. Popovich, Manfred Schrenk, and Kyrill .. Korolenko, editors, *Information Fusion and Geographic Information Systems*, Lecture Notes in Geoinformation and Cartography, chapter 2, pages 18–33. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-37628-6. doi: 10.1007/978-3-540-37629-3_2. URL http://dx.doi.org/10.1007/978-3-540-37629-3_2.

Eija Kaasinen. User needs for location-aware mobile services. *Personal Ubiquitous Comput.*, 7(1):70–79, May 2003. ISSN 1617-4909. doi: 10.1007/s00779-002-0214-7. URL http://dx.doi.org/10.1007/s00779-002-0214-7.

Gerard Kickul and Jill Kickul. Closing the gap: Impact of student proactivity and learning goal orientation on e-learning outcomes. *International Journal on E-Learning*, 5(3):361–372, 2006. ISSN 1537-2456. URL http://www.editlib.org/p/5958.

KSERVER™. Overview: LMS Solution. http://kserver.360training.com/kserver/KSERVER+Whitepaper+Ver1.pdf, 2012. [Online; accessed 01-August-2012].

J. C. R. Licklider. Man-Computer Symbiosis. *IRE Transactions on Human Factors in Electronics*, HFE-1:4–11, March 1960. URL http://memex.org/licklider.pdf.

A. LEWANDOWSKA M. DEBES and J. SEITZ. Definition and Implementation of Context Information. http://www.wpnc.net/fileadmin/WPNC05/Proceedings/Definition_and_Implementation_of_Context_Information.pdf, 2012. [Online; accessed 05-August-2012].

Walid Maalej. Context-aware software, engineering and maintenance: The fastfix approach. Online Slideshow, 2011. URL http://www.slideshare.net/maalejw/context-aware-software-engineering-and-maintenance-the-fastfix-approach#btnNext.

Don McIntosh. Vendors of learning management and e-learning products. June 2012. URL http://www.trimeritus.com/vendors.pdf.

Tudor Girba Adrian Kuhn Adrian Lienhard Oscar Nierstrasz, Marcus Denker and David Rothlisberger. Self-aware, evolving eternal systems. Technical Report IAM-08-001, University of Bern, Institute of Applied Mathematics and Computer Sciences, 2008. URL http://www.iam.unibe.ch/~scg/Archive/Papers/Nier08aSelfAwareEternal.pdf.

Gilbert Probst and Stefano Borzillo. Why communities of practice succeed and why they fail. *European Management Journal*, 26(5), 2008. URL http://EconPapers.repec.org/RePEc:eee:eurman:v:26:y:2008:i:5:p:335-347.

Antti Salovaara and Antti Oulasvirta. Six modes of proactive resource management: a user-centric typology for proactive behaviors. In *Proceedings of the third Nordic conference on Human-computer interaction*, NordiCHI '04, pages 57–60, New York, NY, USA, 2004. ACM. ISBN 1-58113-857-1. doi: 10.1145/1028014.1028022. URL http://doi.acm.org/10.1145/1028014.1028022.

B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, Washington, DC, USA, 1994a. IEEE Computer Society. ISBN 978-0-7695-3451-0. doi: 10.1109/WMCSA.1994.16. URL http://dx.doi.org/10.1109/WMCSA.1994.16.

Bill N. Schilit, Norman I. Adams, and Roy Want. Context-aware computing applications. pages 85–90, December 1994b. URL http://http://schilit.googlepages.com/wmc-94-schilit.pdf.

Denis Shirnin. Proactive learning management systems. [Stored as an internal document at the University of Luxembourg], 2012.

Yufei Shu and Kazuo Furuta. An inference method of team situation awareness based on mutual awareness. *Cogn. Technol. Work*, 7(4):272–287, November 2005. ISSN 1435-5558. doi: 10.1007/s10111-005-0012-x. URL http://dx.doi.org/10.1007/s10111-005-0012-x.

David Tennenhouse. Proactive computing. *Commun. ACM*, 43(5):43–50, May 2000. ISSN 0001-0782. doi: 10.1145/332833.332837. URL http://doi.acm.org/10.1145/332833.332837.

TheFreeDictionary. Wordnet 3.0, farlex clipart collection. Aug 2012. URL http://www.thefreedictionary.com/background+knowledge.

Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991. URL http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

E. Wenger. Communities of practice a brief introduction, June 2006. URL http://www.ewenger.com/theory/index.htm.

E.C. Wenger and W.M. Snyder. Communities of practice: The organizational frontier. *Harvard Business Review*, 78(1):139–145, January 2000.

Etienne Wenger. *Communities of practice :learning, meaning, and identity*. Cambridge University Press, Cambridge, 1998. ISBN 0521430178; 0521663636. URL http://biblioteca.uoc.edu/llibres/14863.htm.

Etienne Wenger. Supporting communities of practice. a survey of community-oriented technologies. Technical report, ewenger.com, 2001. URL http://www.ewenger.com/tech/index.htm.

Etienne Wenger, Richard McDermott, and William Snyder. *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Harvard Business School Press, Boston, MA, USA, 2002. ISBN 1578513308.

Wikipedia. Lazy evaluation — Wikipedia, the free encyclopedia, 2012a. URL http://en.wikipedia.org/wiki/Lazy_evaluation. [Online; accessed 22-July-2012].

Wikipedia. Ocop — Wikipedia, the free encyclopedia, 2012b. URL http://en.wikipedia.org/wiki/OCoP. [Online; accessed 13-July-2012].

Denis Zampunieris. Implementation of a proactive learning management system. In Array, editor, *E-Learn - World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education*, Hawaii ;USA, 2006a.

Denis Zampunieris. Implementation of efficient proactive computing using lazy evaluation in a learning management system. In Array, editor, *m-ICTE - International Conference on Multimedia and Information & Communication Technologies in Education*, Seville;Spain, 2006b.

Denis Zampunieris. Implementation of efficient proactive computing using lazy evaluation in a learning management system. *International Journal of Web-Based Learning and Teaching Technologies*, 3:103–109, 2008.