UMinho|2009

Paulo Manuel Gonçalves de Almeida **Student Assessment System – Diagnosing Student Models**

Universidade do Minho
Escola de Engenharia

Paulo Manuel Gonçalves de Almeida

Student Assessment System – Diagnosing Student Models

Outubro de 2009

Universidade do Minho
Escola de Engenharia

Paulo Manuel Gonçalves de Almeida

Student Assessment System - Diagnosing Student Models

Mestrado em Informática

Trabalho efectuado sob a orientação do
Dr. Paulo Novais

Outubro de 2009

Acknowledgments

Hard work would not be enough to take this project to its end. Nor it would be the delight I took from dealing with this research challenges. Therefore, I'm indebted to many people who helped and inspired me during the development and writing of my MSc dissertation.

However, special thanks go to

My Supervisor, Professor Paulo Novais, PhD., for his ideas, suggestions and encouragement to go forth.

My Father and Mother, for their love and support.

I Gratefully Acknowledge

The Direction and Administration Board of Professional School *Centro de Formação Profissional da Indústria de Calçado, São João da Madeira, Portugal*, particularly Dr. Eduardo Costa and Dr. Carlos Fragão, for their kindness in financing this work and letting me use the school's MOODLE e-learning platform for my research activities.

Resumo

A necessidade de maximizar o sucesso dos alunos e produzir profissionais com o perfil adequado para preencher as necessidades do mercado levanta a questão do seguimento e avaliação dos percursos de aprendizagem dos estudantes das Escolas Profissionais. Para resolver rapidamente problemas e dificuldades que se levantem durante o processo de aprendizagem, precisamos de desenvolver tecnologias e ferramentas que permitam a monitorização dos percursos.

Suportado numa base de conhecimento de características de estudantes, também chamada Modelo de Aluno, e numa base de conhecimento sobre comportamentos de aprendizagem, um Sistema de Seguimento de Alunos deve ser capaz de produzir diagnósticos dos percursos de aprendizagem. Dada a amplitude das experiências e condutas de aprendizagem dos estudantes, o que implica um grande número de atributos e valores nos modelos de aluno, uma tal ferramenta deve possuir algum tipo de inteligência.

Este trabalho apresenta uma proposta de desenho e implementação de um Sistema de Seguimento de Alunos, uma ferramenta de apoio à decisão no sentido em que, em face de um novo problema (um perfil de estudante) sugere uma solução, i.e., um diagnóstico do estado da aprendizagem do estudante. O seu objectivo é auxiliar os professores e outros técnicos a detectar sinais de problemas de aprendizagem num estudante, de modo a tomarem as medidas apropriadas para os ultrapassar.

Este Sistema de Seguimento de Alunos está, de facto, em teste em ambiente real sendo objecto de um caso de estudo onde os diagnósticos do sistema sobre um conjunto de estudantes são comparados com os diagnósticos dos professores sobre o mesmo conjunto de estudantes.

Palavras-Chave: e-Learning, Student Assessment, Student Models, MOODLE, Decision Support Systems, Quality-of-Information.

Abstract

The necessity to maximize the success of the students as well as to produce professionals with the right skills to fulfill the market requirements raises the question of closely following and assessing the learning paths of the students of Professional Schools. To solve at once problems and difficulties that arise during the learning process, we need to develop technologies and tools that allow the monitoring of those paths.

Supported on a knowledge base of student features, also called Student Models, as well as on a knowledge base about learning behaviors, a Student Assessment System must be able to produce diagnosis of students' learning paths. Given the wide range of students' learning experiences and conducts, which implies a wide range of attributes and values in students' models, such a tool should have some sort of intelligence.

This work presents a proposal for the design and implementation of a Student Assessment System, a decision support tool, in the sense that, presented with a new problem (a student outline) it suggests a solution, i.e., a diagnostic of the student learning state. Its purpose is to help teachers and other technicians to detect signs of learning problems on a student, and to take the proper measures to overcome them.

This Student Assessment System is actually being tested in a real environment and is the subject of a case study where the diagnostics of the system upon a set of students are compared with the diagnostics of the teachers upon the same set of students.

Keywords: e-Learning, Student Assessment, Student Models, MOODLE, Decision Support Systems, Quality-of-Information.

Glossary

AI	Artificial Intelligence
AICC	Aviation Industry CBT (Computer Based Training) Committee
API	Application Program Interface
B_c	Beliefs a system has about Learning Behavior
B_s	Beliefs a Student has about her behavior
CBR	Case-Based Reasoning
CF	Certainty Factor
CFPIC	Centro de Formação Profissional da Indústria de Calçado
CMS	Course Management System
CSV	Comma Separated Values file type
DBMS	Database Management System
DSS	Decision Support System
ELP	Extended Logic Program
GUI	Graphic User Interface
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
ITS	Intelligent Tutoring System
LCMS	Learning Content Management System
LMS	Learning Management System
LP	Logic Program
MOODLE	Modular Object-Oriented Dynamic Learning Environment
Prolog	SICStus Prolog
SAS	Student Assessment System
SCORM	Sharable Content Object Reference Model
SLP	Student Learning Path
SM	Student Model
SMB	Student Model Builder
SMD	Student Model Diagnoser
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCO	Total Cost of Ownership
TCP/IP	Transmission Control Protocol/Internet Protocol
TS	Tutoring System
UDF	User-defined Function
UM	University of Minho
XML	Extensible Markup Language

Typographical Conventions

NewsGotT italic	Predicates, attributes, variables
NewsGotT italic bold	Database table identifiers and columns
NewsGotT bold	Chapters, sub chapters, figures identifiers
`Courier`	SQL code
Times New Roman	Formulas

Contents

List of Figures

List of Tables

1 Introduction

The opportunity for this work emerged from the environment of a Professional School, Centro de Formação Profissional da Indústria de Calçado (CFPIC), equipped with the latest technology in education. Within this environment questions started to arise on how to use that technology and the available data to provide information about the students' careers. The conviction that the interactions of the students with the technology had pedagogical usability, at least in pointing out a trend in the student learning process, became more and more firm.

The idea to develop a student assessment system on top of the information technology already available grew stronger. CFPIC decided that the idea worth a try and realized that, to make it happen, it should acquire know-how. That's how this MSc came along.

In this first chapter, we'll introduce the motivation and scope of this work as well as the key notions of e-learning systems and student models. We'll also define the objectives of student assessment and describe the structure of the work.

1.1 Background and Motivation

Professional Schools, sometimes called Vocational Schools, were born to provide the production system that stemmed from the Industrial Revolution with qualified workers for the different activity sectors and industries. These structures were frequently created outside of the traditional Education System but linked to industrial clusters. Their student population was quite homogeneous in its social backgrounds and expectations [1], and the knowledge available was relatively narrow and stable[1]. In this

[1] The contents of the students' Shoes Sewing Manual had nothing but minor changes during the time frame of over twenty years (source: CFPIC library).

environment, it was easy to control the process of teaching and learning and to guarantee the success of the students, as well as to fulfil the enterprises' requirements of skills.

Nowadays, these schools are immersed in an information society, confronted with new technological challenges, and with a student population that comes from different social backgrounds, carrying with it different needs and expectations. Moreover, the enterprises are pressing for more qualified technicians, able to help them embody this technological revolution to keep up with the global competition. Schools are, therefore, faced with a new technological paradigm, a new kind of public and new demands from the enterprises.

Professional Schools have tried to cope with these challenges by investing in organization, management, market research, and in human and technological resources, as well as in new pedagogical tools, such as computer systems, multimedia projectors, interactive whiteboards and e-learning platforms.

These investments are very expensive; schools cannot afford to have unsuccessful students. As a consequence, the students' careers must be closely followed. Therefore, schools should have devices to evaluate their students' learning state, i.e., they should possess means to keep their students' descriptions up to date, that way being able to follow and diagnose their learning paths, if not in real time, at least periodically, to avoid failures as much as possible. Furthermore, the need to supply the market with effectively qualified personnel favours these evaluations.

This evaluation and following should be performed by teachers and psychologists, who access and diagnose the learning paths of the students to detect symptoms of deviations, decide the appropriate "therapies" and act accordingly.

1.1.1 Information Technology in the Classroom – the Price of Failure

During the last decade or so, chiefly with the advent of the 21th century, schools have been spending increasingly more money to incorporate technology in the classroom.

This revolution in learning is occurring for many reasons. Increasingly, parents are demanding that their children have access to the latest technology; markets are demanding for more qualified technicians with solid knowledge on Information Technology (IT); school officials want to make their schools more attractive to the social community; and politicians, well, politicians are responding. Governments are freeing more funds to support technological improvements. And there is a growing consensus on the idea that, if technology is wisely incorporated, it can improve the learning experience.

In Portugal, the “Plano Tecnológico da Educação” [2] program, created in 2007, has provided an infusion of funds to help public schools of the 2nd and 3rd levels get wired and connected to the Internet, to equip them with interactive whiteboards and projectors, and to distribute computers to the children of the first degrees of education (the Magalhães PC). Furthermore, the “Novas Oportunidades” [3] program is helping people with low qualifications, which were outside the education system (in some cases for many years), to return to the school, and is financing their acquisition of computers with wireless Internet access. Many of these new students were unemployed and came to Professional Schools.

So, Professional Schools are not apart from this trend of taking IT to the classroom. In fact, they have the same social and technological pressures of public schools (though, often, fewer funds).

1.1.2 The Total Cost of Ownership of Classroom IT

When a school purchases computers or installs a network, the costs of the hardware, software and installation services are only a small part of the expenses it can expect in subsequent years, if it is going to use those technological resources effectively. After a school has purchased computers, projectors, whiteboards, installed a networking infrastructure with internet connection, and set up pedagogical software tools on top of it, there are expenses for which school administrators must be prepared [4]. Namely:

- **Retrofitting**: when the school is ready to build a network, it must be sure that it has budgeted adequately the electrical capacity upgrade, the heating, cooling and ventilation systems improvement, the security systems expansion, etc. These costs can be reduced if a school plans for future networking requirements when its buildings are constructed or renewed. Unfortunately, there is a lack of forecast talent in many school boards. Nowadays, though, wireless solutions may offer an option to wiring with cost savings, although with some drawbacks (security, performance, etc.).

- **Professional Development**: the school must budget an adequate amount for staff training, including the cost of trainers, materials and substitutes if training is conducted during school hours. Training costs can represent a large component of a school's technology budget. If staff members are not properly trained, teachers will not understand how to integrate technology into the curriculum, support staff will not keep up with hardware and software developments and concomitant problems and solutions; and the school will fail to achieve the maximum return on its technology investment.
- **Software**: the school should budget for network management software, computer based curriculum materials, applications and productivity software and e-learning platforms. A wide range of software applications will give schools greater flexibility, but will also increase the costs for support and staff development. Software licenses also need to be managed efficiently to save money and protect the school from penalties for license violations. Open source software should be considered whenever a choice is possible.
- **Support**: the school must compute the costs to maintain its network and other hardware and software systems, and the costs to help their staff and students get their software and hardware problems solved. The way in which a school deploys a network and the variety of software and operating systems that it chooses to support, will determine the kind of support that it will need.
- **Replacement Costs**: the school has to budget adequately to cover the costs of replacing computers and other peripherals. And to periodically renew their set of software licenses.
- **Connectivity**: the school has to budget to cover the costs involved with connecting to the Internet. Lower-bandwidth connections will cost less but will have a tradeoff to pay in the complexity of the information that can be shared and the amount of time it will take to download files or access information.

With this in mind, we can see that the Total Cost of Ownership (TCO) of technology per student can be high. In 1995 McKinsey & Company, Inc. published a study [5] where the costs to equip schools with IT and internet connections were estimated. That document considered several scenarios of implementation of IT in school environment, namely:

- The **Classroom Model**, in which every classroom is connected with networked computers at a ratio of five students per computer, with a T-1 (1,54 Mbps) connection permitting long-distance transmission of data, video and voice. The model considered onetime costs per student and ongoing costs per student per year over 10 years.
- The **Partial Classroom Model**, in which only half of each school's classrooms are wired. The costs considered are: onetime costs per student and ongoing costs per student per year over five years.
- The **Lab Model**, which assumes each school is connected through a computer lab of networked computers with 10 analogue telephone lines per school. The model considered onetime costs per student and ongoing costs per student per year over five years.

Using the same model, but upgraded to today's technology and costs, one can estimate the initial costs and the ongoing costs of the partial classroom model for a school with 30 classrooms and 500 students, and assuming 15 classrooms with 20 computers for the students and every classroom with one computer for the teacher. In fact this is roughly the scenario of the CFPIC Professional School. The estimation is based on Portuguese market prices in January 2009 for the referred equipment. **Table 1** shows the results.

The 1st year costs include the investment and the first year operation. In the acquisition of an e-learning Platform we're only considering the price of the server hardware and operating system that hosts the platform, as well as the installation and configuration of the platform; the platform itself is considered free. This is because we're assuming that the one chosen will be open source and free of charge. If this is not the case, that is, if the e-learning platform is not free, the initial costs will rise to higher figures.

As for the ongoing yearly costs, it is assumed the value of 15% of the initial costs for maintenance contracts on hardware and software. The ongoing cost of the platform is only the cost of the human resources that customizes it to the school environment and needs, and that maintains it running. Obviously this is a simplification because we're not taking into account the costs of developing the courses, the evolution of memory and disk usage, the costs of the supporting infrastructure (energy, heating, cooling, etc...), the costs of staff training, etc. Even so, one sees that these are huge investments.

Should we include cabling, switching and routing, and, in general, backbone costs, and the figures would be grater. Indeed, considering only the costs of introducing IT in the classroom, there is a big price to pay for student failure.

Item	Qty	Unit. Price	Investment	1st Year Cost	Yearly Cost	1st Year Cost/Student	Yearly Cost/Student
Computers	330	600,00	198.000,00	227.700,00	29.700,00	455,40	59,40
Internet Connections	1	1.300,00	1.300,00	19.300,00	18.000,00	38,60	36,00
Projector	30	650,00	19.500,00	22.425,00	2.925,00	44,85	5,85
Interactive Whiteboard	30	1.000,00	30.000,00	34.500,00	4.500,00	69,00	9,00
Productivity SW Licenses	330	50,00	16.500,00	18.975,00	2.475,00	37,95	4,95
e-learning Plataforms	1	6.000,00	6.000,00	21.000,00	15.000,00	42,00	30,00
Total			271.300,00	343.900,00	72.600,00	687,80	145,20

Table 1 Estimation of the Costs in € of IT in Classroom, 2009

1.2 Scope of this Work

So, Professional Schools are faced with this problem of having to avoid student failure. How can they be helped? One possible answer is to provide them with tools to help assess and follow their students' careers. Such tools should detect in time learning problems and difficulties. As soon as these problems are detected, the school can devise strategies to solve them.

This work presents an approach to design and build a Student Assessment System (SAS) for Professional Schools [6]. A SAS is indeed a tool to help teachers, psychologists, tutors, i.e., the decision makers, to assess and evaluate the students' careers. Supported on students' features, or Student

Models, and on knowledge about learning behaviors, a SAS must be able to follow and produce diagnosis of the students' learning paths.

The SAS uses data from several sources to build student models. Among those sources, the SAS uses data from the interactions of students with e-learning platforms to evolve Student Models and derive its knowledge about learning behavior. Based on those models and on that knowledge it presents the decision makers with a diagnosis of the student learning state.

1.3 E-learning and E-learning Systems

Governments in advanced countries realized the importance of knowledge based economies and this is not another political hype. We can observe this trend everywhere, including in our own country with the aforementioned "Plano Tecnológico da Educação", "Novas Oportunidades", etc.

"Unable to compete with low labor costs in developing countries, more advanced economies are trying to create highly productive (and high wage) knowledge based industries, such as computing, telecommunications, financial systems, and education itself. Such industries depend on a highly educated workforce, thus leveraging an advantage over less economically advanced countries.

Governments see two distinct roles for e-learning. They see e-learning as a product of a new knowledge based industry that can be sold worldwide. At the same time, they see e-learning as a way to improve the quality of education and to produce technology savvy professionals, able to use new technologies in the new economy.

Business also sees a value in e-learning as a way of increasing competitiveness through ensuring that the work-force is continually learning and improving without the high costs of travel and time away from work".

These words, taken from the book **Technology, e-learning and distance education** [7], perfectly describe the strategic role of e-learning in the first world economy.

But what is e-learning? And what are e-learning systems?

1.3.1 E-learning

There are several definitions of e-learning. For instance, Wikipedia ([8]) grabs ideas from a number of sources and comes up with the definition of e-learning as “a general term used to refer to technology-enhanced learning”. Similarly, Nagy ([9]) defines e-Learning “as an umbrella term describing any type of learning that depends on or is enhanced by online communication using the latest information and communication technologies”. Mark Nichols ([10]) is more precise, when he defines e-learning as "pedagogy empowered by digital technology”.

As we can see, the e-learning concept is not stable and appears to mean different things to different people. But, for this work’s sake, we can agree on a definition of e-learning as a merge of Nichols and Nagy concepts: **e-learning is pedagogy empowered with Information and Communication Technologies and tools**. With this definition, we have e-learning as another pedagogical tool, yet one of the most powerful.

1.3.2 E-learning systems

E-learning systems are software programs that provide support for learning activities. They include personal training systems, usually designed for a certain knowledge domain, as well as general learning management tools suitable to manage and deliver distinct types of learning content, covering several knowledge domains.

A personal training assistant, who pays attention to the student's learning needs, assesses and detects problems, and provides assistance as needed, is called a Tutoring System. The Tutoring System possesses expertise about the knowledge domain.

The Andes Physics Tutor [11], first used in the United States Naval Academy, is an example of a Tutoring System. It is designed to be an intelligent homework helper for physics. “It replaces the pencil and paper that students would ordinarily use to solve physics homework problems. Students draw diagrams, enter equations and define variables with the same freedom that they have when using paper. Yet, unlike a piece of paper, Andes tells students whether their entry is correct by turning it red or green, and Andes will give principle-based hints when asked”. The underlining is mine, stressing some functionalities of a Tutoring System.

The Learning Management Systems/Learning Content Management Systems (LMS/LCMS), on the other hand, are domain independent, general purpose programs/platforms, which provide authoring, sequencing, and aggregation tools that structure content to ease the learning process. It is the responsibility of the course designer to select and organize the matter in order to build a pedagogical module for a knowledge domain.

MOODLE platform [12] (Modular Object-Oriented Dynamic Learning Environment) is an example of a LMS/LCMS. It is widely used and the number of installations is growing at a fast pace. It is also the platform we use in this project.

1.4 Student Models

Whatever the case, Tutoring Systems (TS) or LMSs, systems should have some sort of knowledge about the student and about her learning process. That is, the system must possess a set of representations about the students, their behavior and their knowledge in order to adapt to each students' characteristics. For instance, MOODLE must know the next task to deliver to a student that matched a question in a lesson activity; and Andes must know what hint to provide in the face of a certain step of a given task. This knowledge the system has about the student, is usually called the Student Model (SM). The SM allows the e-learning system to adapt its presentations to the student. Without a SM a system would simply behave the same way for all students.

Student modeling is a special case of user modeling. Its level of complexity (and, therefore, that of the resulting SM) varies, being simpler in LMS (a little more than a student profile, in most cases) and more complex in ITS.

1.5 Related projects

E-learning is a vast and highly populated research area, so to speak, where many concurrent projects are being developed, covering many topics. For this work many of them were important. The following ones were most valuable for us.

The MSc dissertation of Manuel Rodrigues [13], being a survey of Tutoring Systems field, describes many projects in Intelligent Tutoring arena, and presents a framework for development of Intelligent Tutoring Systems.

The works of Romero and his team ([14], [15], [16]) around the problems of extracting association rules from e-learning systems were very inspiring and gave us some worthy ideas for this work, chiefly on the process of initialization of the student models and the knowledge base.

The research of Calvo-Flores and others ([17]) on the issues of predicting students' marks from MOODLE logs was very interesting to us, although they've used artificial neural networks to model students behavior, and we have used an extended form of predicate logic (Extended Logic Programs).

1.6 Student Assessment Problem Definition

For the matter of this work, assessment is the problem of determining what chances a student has of passing a certain course, just looking at the features of the SM. It is, in fact, performance judgment.

A SAS, as the one that will be described, configures a Decision Support System which relies on an e-learning platform to acquire data about the students' interactions with the learning domains, as well as their engagement on the activities proposed, their performance on examinations and quizzes, etc. This SAS is expected to be able to initialize the SMs with foreign data about socio-economic background, former academic history, etc. With this information, i.e., the interactions and the personal information, the SAS updates the SMs and evolves its knowledge about students and about learning behaviors; eventually, it will output a diagnosis about the students' learning paths.

With reliable evaluations and diagnosis of their performance, students, teachers, and school boards can improve the distribution of pedagogical resources, adapt those resources, increase help to some students, in short, detect and solve problems as they arise.

1.7 Hypothesis

Our guess is that the interactions of the students with an e-learning platform have pedagogical utility, and can be used to point out a trend in the student learning process, making it possible for teachers and other decision makers to detect problems on that process sooner.

1.8 Objectives

The main objective of this work is, therefore, to design and develop a system that can help the schools assess, diagnose and follow the students' learning paths based on the information derived from their interactions with an e-learning platform. In order to achieve this objective, the following goals were set:

- Define a Student Model suitable to express the student set of relevant characteristics to produce diagnosis of student learning paths;
- Define an adequate and effective representation of system knowledge able to express incomplete information and to assess the Quality-of-Information of Student Models;
- Define a conceptual framework to support the design and developing of the system;
- Design an architecture for the system;
- Develop a prototype of the system.

1.9 Research methodology

Like many researchers nowadays, we have used Action Research as the selected method of problem solving. Generally, we can define Action Research as "learn learning by doing" - a group of people identify a problem, do something to resolve it, see how successful their efforts were, and if not satisfied, try again [18]. This methodology is comprised of the following steps:

- Specification of the problem;
- Data collection about the problem and the problem domain;

- Postulation of possible solutions for the problem followed by a definition of a course of action;
- Taking of the actions proposed;
- Evaluation of the effects of the actions taken.

At this point, the problem is re-assessed and the process begins another cycle. This process continues until the problem is solved.

1.10 Structure of the Present Document

We'll start by summarizing the notions of e-learning systems and their architecture, present some examples as well as some features of MOODLE e-learning system that are of interest for this work, and where the SAS fits in that architecture.

Next we'll formalize the concepts of student model, student modelling and belief; in this context, we'll also address the problems of measuring the Quality-of-Information of the student models, the representation of knowledge and the definition of the theoretical foundations and basic framework.

Then the architecture of the SAS, followed by its components' operation, will be presented. After that, the processes of initializing and operating the SAS will be described in the context of a case study.

Finally, we'll discuss the ideas and the results of this work when applied to real situations, and bring out some lines of thought and research for future work.

All chapters start with an introduction that presents the subject of the chapter and end with a conclusion that sums up the contents of the chapter. This first chapter is intended to give a summary of this work.

2 E-learning Systems

In the previous chapter a definition of e-learning as pedagogy empowered with Information and Communication Technologies was presented. And e-learning systems were described as software programs that provide support for learning activities. This chapter will address the general architecture of e-learning systems and the two major flavors of such systems. It ends with a short analysis of MOODLE LMS: Its objective is to provide an understanding of the various elements that make up e-learning systems and a general view of MOODLE e-learning platform.

2.1 E-learning Systems' Design Principles

The architecture of an e-learning system follows the principles of today's systems' architecture design. Generally,

- Interoperability;
- Scalability;
- Globality;
- Integration;
- Flexibility;
- Stability, High availability and Performance;
- Security;
- Usability.

2.1.1 Interoperability

Support of content from different sources and multiple hardware/software vendors' solutions. Support of established industry standards, to allow addition of future modules; especially it should be based on open industry standards for Web deployments (XML [19], SOAP [20]). And support for the major learning standards (AICC [21], SCORM [22], IMS [23] and IEEE [24]).

2.1.2 Scalability

The system's infrastructure should be able to expand to meet future growth, both in terms of the volume of instruction and the size of the audience.

2.1.3 Globality

It must serve the largest possible base of end-users, with both synchronous and asynchronous distance delivery of products/contents.

2.1.4 Integration

The system must have front-end and back-end connections to organizational resources. It must not be an island inside the organization (school, business or else).

2.1.5 Flexibility

Openness to emerging technologies and use of new best practices it's a precondition for the system's success.

2.1.6 Stability, High availability and Performance

The e-learning infrastructure must be robust enough to serve the needs of hundreds of learners, administrators, content builders and instructors simultaneously. It must be reliable enough to effectively manage a large implementation running 24 hours a day, 7 days a week. Its products/contents must reach the users in time, anytime, anywhere.

2.1.7 Security

As with any collaborative solution, e-learning systems can selectively limit and control access to online content, resources and functions, both internally and externally, for its user community.

2.1.8 Usability

The access, delivery and presentation of material must be easy-to-use and highly intuitive — just like surfing on the Web or shopping at an online store.

2.2 E-learning System's Architecture

The technology that best adheres to these principles is the Web-based application technology[2] and, for that matter, is the technology most widely adopted to develop e-learning systems. The next figure illustrates the architecture of an e-learning portal site (Intranet or Internet application).

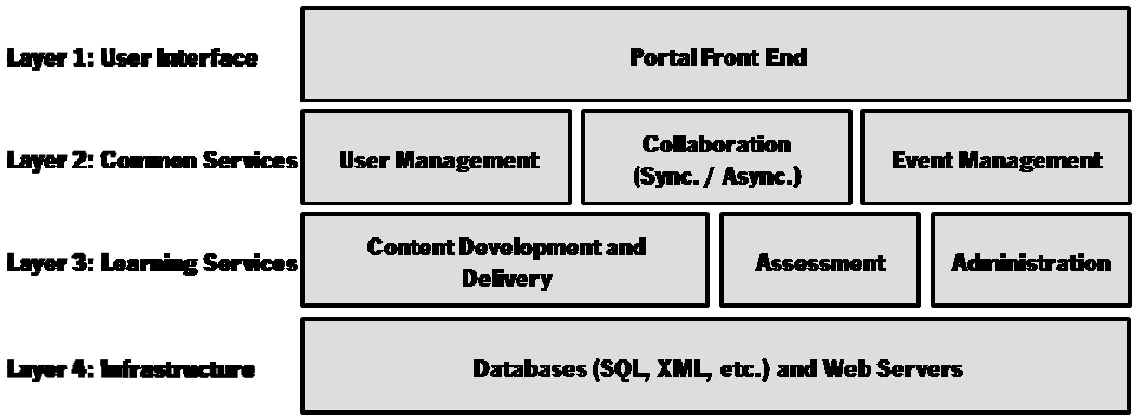


Figure 2-1 General e-learning system architecture (based on [25])

This architecture assumes the usage of web technologies to deliver and manage e-learning content. The system is composed of four layers:

Layer 1 - User Interface: allows users to access the platform via a web browser;

Layer 2 - Common Services: provides services needed by every user and not linked to any specific pedagogic function or activity;

Layer 3 - Learning Services: provides modules for the production and delivery of e-Learning resources; provides, as well, modules for assessment of the users and administration of the platform;

[2] A Web-based application is one that is accessed via a web browser over a network such as the Internet or an Intranet. It uses browser supported programming languages and relies on the web browser to render its "executable".

Layer 4 – Infrastructure Services: provides infrastructure services to store all data related to the system (user interaction, content handling, administration and management, etc.), perhaps interconnected with XML database technology to ease presentation and navigation of the content; this layer supplies, as well, client-server services using standard internet protocols.

Let's look deeper into each layer to see the functions and services that an e-learning system must possess and make available.

2.2.1 Layer 1 – User Interface

The User Interface is composed of an **Internet/Intranet portal** site. An Internet or intranet portal provides a single point of access for all users, via a web browser. Some pages, informational or institutional, for example, may be open for anyone to see, while access to higher levels of service may be restricted by a log-in.

2.2.2 Layer 2 – Common Services

The Common Services include User Management, Collaboration and Event Management.

- **User Management**: the User Management service identifies and assigns privileges to every user of the system. Each user is identified by a unique ID, to which roles can be assigned. Each role defines the privileges of the actors of the system: students, teachers, administrators, authors, etc. Roles and privileges can be changed as needed. The User Management service records and handles all of this user information and conducts the authentication process.
- **Collaboration (Synchronous / Asynchronous)**: the Collaboration service provides communication among the users. Synchronous collaboration technologies include: virtual classroom (with audio-visual, and whiteboard resources), virtual meeting rooms, and chat. Asynchronous collaboration technologies include: email, threaded discussions, and peer-to-peer instant messaging.
- **Event Management (Calendar/Scheduling/Reminders)**: students need to be able to see their enrollments, their work assignments, the events of the school, etc.; students and teachers need reminders to support their workflow, etc.

2.2.3 Layer 3 – Learning Services

The Learning Services include such systems as Learning Content Development and Delivery, Assessment and Administration.

- **Content Development and Delivery**: there are, roughly, two kinds of Learning Content Development and Delivery Systems: the Learning Content Management System/Learning Management System (LCMS/LMS) and the Tutoring Systems (TS), as we've seen in last chapter. These systems will be described on the next sections.
- **Assessment**: this service provides authors/instructors with resources to generate quizzes and tests. Typically this service supports selection of questions from a growing question pool, and automation for generating online tests and quizzes that will be automatically scored for the instructor.
- **Learning Administration**: the Learning Administration service allows back office management of curriculum, resources, teachers and students. It also provides templates and resources for automated reporting. In some cases, this service is tied to e-commerce functions for efficient management of fee payments and related administration functions.

2.2.4 Layer 4 – Infrastructure Services

This layer is composed of the database engines (SQL [26], XML) for data store and retrieval. It has, as well, services to provide support for the distribution of the content material, and services to handle all the users' interaction over the networks. It uses standard protocols like TCP/IP [27] and HTTP [28].

The data generated by the first three layers is very valuable for a SAS. Level one is, obviously, important because it is where the student interacts with the system. Layer two allows us to track the collaboration interactions and the work done without assistance by the student. In layer three, the assessment service is of paramount importance because of the management of quizzes and other tests, which allows for the record of the approval level of the students. But layer four is the most important level for the operation of a SAS as the one that is the subject of this work. All data produced by the other layers is stored in layer four. For a SAS like this it suffices to interact directly with the databases of the e-learning system to have access to the information it needs. It reads the information produced by the other levels to evolve its knowledge about the students, as we've briefly said in the first Chapter and will show deeply later on. And it can also write the knowledge it had learned in layer four databases.

2.3 ITS

As said before, a TS has knowledge domain expertise. It has also a representation of the students and a student modeler which is able to assess the student representation and create an image of the student knowledge state. With that information, the tutoring module can adapt the learning process to the student's needs (**Figure 2-2**).

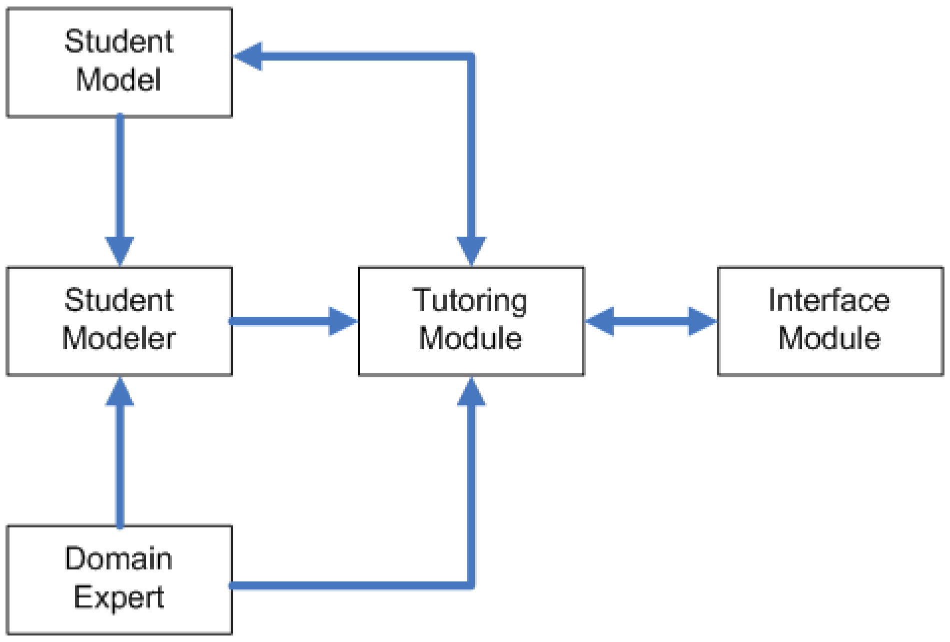


Figure 2-2 High level view of the modules of an ITS

By analyzing tutoring as a set of tasks, each of which can be solved by multiple steps, Van Lehn [29] have divided the duties of a Tutoring System in two loops: an outer loop over tasks, and an inner loop over steps. The main duty of the outer loop is to select a task for the student which will help her learning. The inner loop is responsible for both deriving the right steps from the student behavior, and helping the student to learn. The following algorithm depicts these loops:

```
Until tutoring is done, do:
    Tutor poses a task
    Until the task is achieved, do:
        Tutor may give a hint
        Student does a step
        Tutor gives feedback on the step
    Student submits the solution to the task
```

Generally speaking, the outer loop sequences the tasks to be presented to the student, and the inner loop adapts and helps the student to accomplish the tasks. Not all TS have two loops; some lack an inner loop. Those that only have the outer loop, assign the student a task, collect the student answer, give feedback on the answer and continue, either by giving the student a new chance to answer, or by presenting the student with a new task. Systems that lack an inner loop are called Computer Aided Instruction systems or Computer Based Training systems, and systems that do have an inner loop are called Intelligent Tutoring Systems (ITS).

2.4 LCMS/LMS

Put simply, "a LCMS is a multi-user environment where learning developers can create, store, reuse, manage, and deliver digital learning content from a central object repository" [30]. On the other hand, the LMS interacts with the students, allows catalogue review, course selection and enrollment, student tracking, and launches the online content. Whereas a LMS manages the processes surrounding learning, a LCMS manages the process of creating and delivering learning content.

The LMS is tightly connected to the User Management services in order to give access to resources based on privilege levels, and to establish students tracking (their enrollments and scores). The LMS is able to generate reports of the students' activities and progress towards grading and certification. The LMS not only manages delivery of online content, but is also able to manage students', teachers' and supervisors' schedules and learning events — including room and equipment allocation, chat meetings, quiz and test appointments, etc.

The LMS is the engine behind all generic e-learning platforms.

2.5 MOODLE E-learning Platform

MOODLE is an Open-Source LCMS/LMS (or Course Management System (CMS), as their developers call it). MOODLE is widely used by educators and learners around the world to support their learning communities. Next figure shows the evolution of MOODLE based sites since 2003.

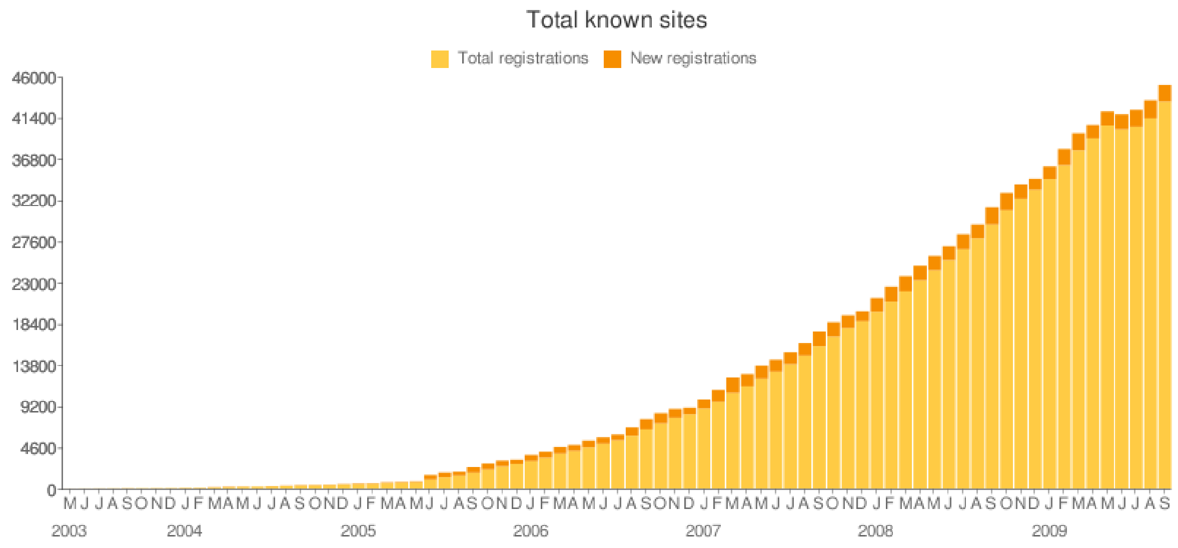


Figure 2-3 Evolution of the number of MOODLE-based sites (source moodle.org)

In the words of people at moodle.org, the design and development of MOODLE is guided by a "social constructionist pedagogy", an educational theory derived from Constructivism. The analysis of the truth of this claim is out of scope of this work. Constructivism is a theory of knowledge developed by Jean Piaget, which argues that human beings learn by experience. Social Constructionism extends the notion of Constructionism [31] to social environments, defending that groups construct reusable knowledge for one another by collaboratively exchanging objects and meaning. Anyway, MOODLE, as said before, is a LCMS/LMS built upon the concept of collaborative learning[3]. It is a free Open Source software package designed to help educators by creating effective online learning communities.

[3] As opposed to ITSs, which are mostly individual learning environments.

2.5.1 MOODLE Overview

Development of MOODLE first begun with Martin Dougiamas, who wished to get away from WebCT (now Blackboard [32]) and create a new and easier alternative. Since then, he launched the first official release of MOODLE (version 1.0) to the public in August 2002, and ever since, the project has evolved greatly, with contributors from all around the world adding new features, better scalability and improved performance. MOODLE is now in the version 2.0.

One great strength of MOODLE is that it is platform independent, meaning it can be run on UNIX [33], Linux [34], Windows [35], Mac OS X [36], or any other system which supports PHP [37]. Its data is stored in a single database, which is generally either a MySQL [38] or PostgreSQL DBMS [39], though it can also run with Oracle [40], Microsoft SQLServer [41], ODBC [42] and others systems, through use of its ADODB technology.

MOODLE is very simple to install on any platform that supports PHP. It is lightweight and compatible with almost any browser interface. Courses can be easily created by a teacher or administrator, and these can be categorized and searched at will. It is a secure environment where all forms are checked, all data validated, cookies encrypted etc. Furthermore it is very simple to use and inexperienced users can create new courses, or add their own content in an easy and intuitive way.

There are multiple site management options provided for the administrator including

- Plug-in themes, to alter the site appearance;
- Plug-in activity modules, which can be added to existing MOODLE installations;
- Plug-in language modules for numerous types of language localizations.

There are many user management options including a range of different authentication mechanisms, from manual user account creation, to automatic email-based authentication. There are also multiple course management options including

- A choice of course formats, such as by week, by topic etc.;
- An extensive range of course activities, which can be added to at any stage;
- User tracking and logging facilities etc.

And there are multiple other options and features. For these reasons and others, MOODLE is constantly increasing in popularity amongst academic and commercial institutions and is expected to do so even more in the future.

2.5.2 MOODLE Components

MOODLE has a number of interactive learning activity modules like forums, chats, quizzes and assignments that helps learning from a participative position. The teacher (or the course designer) creates and edits the content of the modules and adds pedagogical resources to it. MOODLE also allows for the creation of complex quizzes and tests of any kind. It supports a whole set of question types from simple true/false to multiple choice questions, with one or more answers to completely open answers. Furthermore, MOODLE has a lesson module where it is possible to write interactive learning programs:

- the designer interchanges matter and questions;
- MOODLE is capable of, not only analyzing the results and award points, but also enabling the embedding of paths and conditions;
- depending on how well a student answers, the whole action of the lessons will change.

We can think of this module as an outer loop implementation in the way defined above. Figure below shows a generic perspective of the MOODLE components.

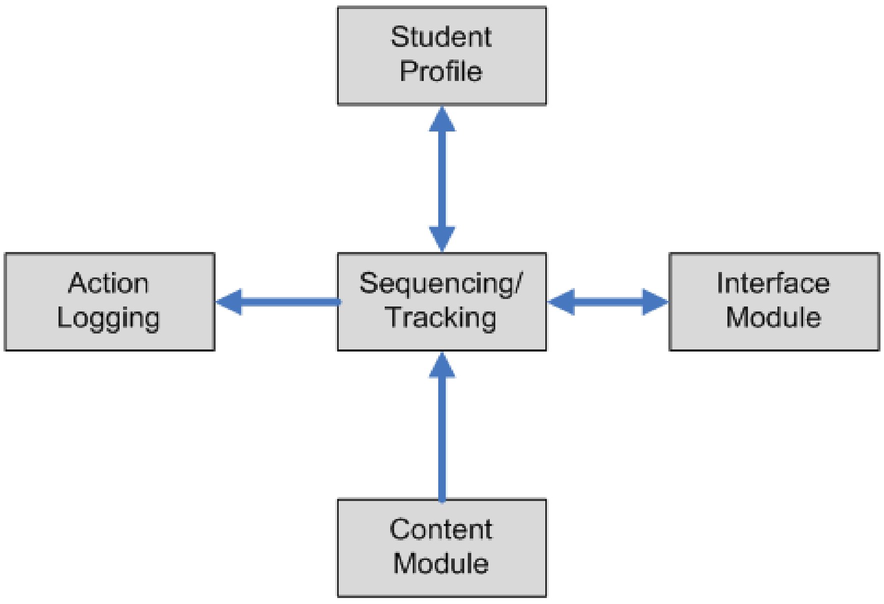


Figure 2-4 Generic view of MOODLE LMS components

In addition to these learning modules, MOODLE includes a logging module to track users' accesses (user identification, IP and time of access) and the activities and resources that have been accessed. Each user (student or teacher), performs an Action at a time on a specific Resource of a given Course. A Resource can be a Label, a TextPage, a WebPage or a Link. A course may have several activities. Each activity is either a Forum, an Assignment, a Quiz or a Lesson. From this log and from the tables that store each activity's data, MOODLE is able to generate, for each student, reports with details about when and how a resource has been visited, the duration of the visit, etc. Administrators and teachers can extract reports from this data.

Furthermore, MOODLE is able to provide a summary about the participation of all students. This information can be combined with biographical, socio-economic and cultural data as well as academic history in order to obtain more complete information about the development of the course and the evolution of the students. Next figure presents the information model around MOODLE activity log.

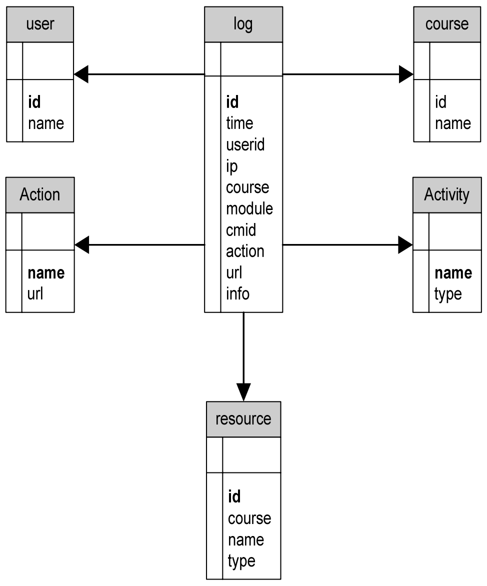


Figure 2-5 Simplified excerpt from MOODLE Information Model

2.6 Conclusion

The SAS relies on the data stored on the fourth layer (database and infra-structure layer) of the e-learning systems' architecture to build and evolve its knowledge about the students. ITSs and LCMSs/LMSs are the two most common implementations of the e-learning systems' concept, being the former linked to a knowledge domain, and the later unlinked to a particular learning domain. In this work we use the MOODLE LMS database as the source of information to the SAS.

3 Student Modeling and Representation

User-modeling is a sub-field of computer ergonomics in which cognitive models of human behavior are developed, including those of skills and knowledge. User models can be used to predict conduct under stress (military), learning state and subsequent evolution, shopping profiling, etc. User models can also guide user interface designers to improve the human-computer interaction.

User modeling when applied to e-learning is called Student Modeling.

This chapter intends to describe student models, their necessity, their contents and their representation under the form of beliefs about the student behavior.

We'll start by explaining the concepts of student model and student modelling, including the SM of the SAS. Then the representation of the knowledge will be described and a way to assess the Quality-of-Information (QoI[4]) of a SM will be presented. Finally, the framework, as well as the concept of belief, and its representation and storage, will be depicted.

[4] Sometimes denoted as QI.

3.1 Student Models

As already said (**1.4**), a student model is a representation of the students in terms of the characteristics (knowledge, personality, background, etc.) which an e-learning system believes that a student possesses.

Student models are found as a component of e-learning systems. E-learning systems must possess a set of representations about the students, their behavior and their knowledge in order to adapt to each student's characteristics, on one hand, and to express her learning state, on the other. That is the main role of a SM in the context of an e-learning system. In short, the models intend to represent properties of the students:

- Their likings and dislikings;
- Their goals and interests;
- Their domain knowledge;
- Their attitude towards the school;
- Their learning behavior;
- Their experiences and backgrounds;
- etc.

What information the student model should be designed to contain depends on the e-learning platform and, of course, the objectives of modeling. Student models have always to be considered in the context where they will be used and the goals of the e-learning system. There are several types of SMs, as we'll see, some of them suiting better to model students on ITSs, and other suiting better to model students on generic LMSs.

In the context of this work, the attributes selected to compose the SM should be meaningful to produce a diagnosis. Biographical, social, economical and cultural data, as well as skills so far achieved, are known constraints to the acquisition of new knowledge. In addition, one needs to select metrics that will allow inferring the quality of the ongoing learning, i.e., the degree of engagement and expertise on the currently attended learning domains.

Socio-economic and cultural attributes, as well as former academic data, are called **static** attributes throughout this work; metrics that come about from the interaction with a learning platform are called **dynamic** attributes.

To collect these attributes' values one needs a learning platform, such as a Learning Management System (LMS) or an Intelligent Tutoring System (ITS), or a combination of these. We use MOODLE LMS technology as we've said. Together, constraints and metrics compose the predicates of SM.

3.2 Student Modeling

Student modeling is a special case of User Modeling. Following Foster and Fletcher [43], the current state of cognitive modeling considers three types of models:

- Implicit Cognitive Models, where the system builds its assumptions about the student based on the answers a student gives to the problems presented;
- Explicit Cognitive Models – Quantitative approach, where the system builds the SM based on the probability of the student to know a X+1 piece of knowledge, having knowledge of X;
- Explicit Cognitive Models – Qualitative approach, where the system models its knowledge of the student using "if-then" rules that simulate cognitive and/or behavioral structures and processes.

3.2.1 Implicit Cognitive Models

Implicit Cognitive Models are lesser used today than they used to be. Generally, when faced with a given answer, the system will choose a certain path. For example, the system assumes that the student knows the multiplication algorithm because she chooses the answer 15 to the question: "What is the result of 3 X 5?". To set up an instructional item, a developer must anticipate and prepare responses for several discrete cognitive states, represented by the right and the wrong answers to the item. The cognitive model represented by these states is static, linked and limited to the problem. That's why developers searched for types of modeling more explicit, i.e., where the model is more abstract and separated from the problem domain.

3.2.2 Explicit Cognitive Models - Quantitative

The second type of models, Explicit Cognitive Models of quantitative type, is being used for applications such as Intelligent Tutoring systems (ITS). With this models, the developer attempts to find the probability that a particular knowledge item, for a given student, goes from unlearned to learned state. The parameters are estimated for each item-student combination. These models use Bayes' theorem [44] to determine the probabilities that a student is in a specific cognitive state. For instance, the system may believe that a student knows the multiplication algorithm because:

1. it believes that she knows how to multiply two single digit factors (multiplication table); and
2. it also believes that she knows how to sum those factors.

The next figure shows the Bayesian model that illustrates these beliefs.

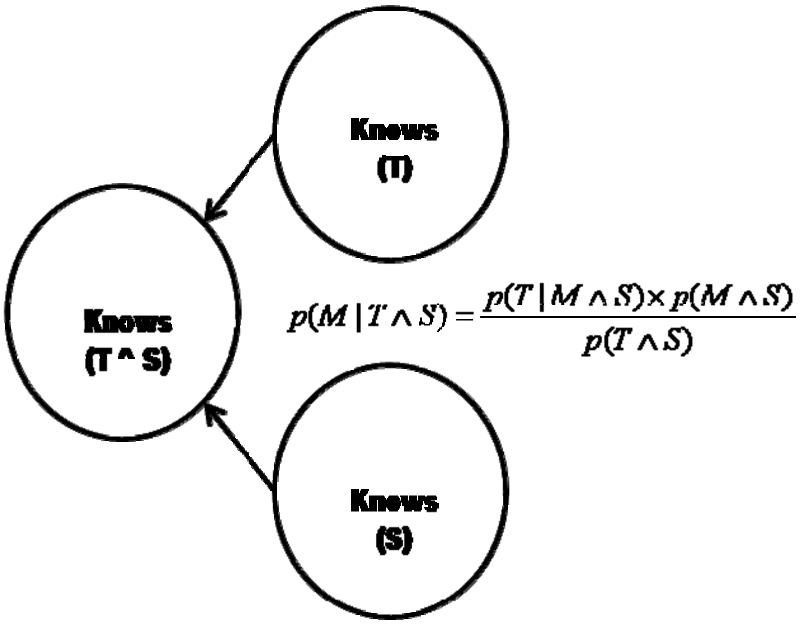

Knows (T)
Knows (T ^ S)
Knows (S)
$p(M \mid T \wedge S) = \frac{p(T \mid M \wedge S) \times p(M \wedge S)}{p(T \wedge S)}$

Figure 3-1 Bayes model for the multiplication algorithm belief

The equation in the figure measures the dependency between knowing the multiplication table (T) and the sum (S), and knowing the multiplication algorithm (M).

3.2.3 Explicit Cognitive Models - Qualitative

The last type of models is not numeric. Rather, it describes objects and processes in terms of spatial, temporal and causal relations. Clancy [45] defines a qualitative SM as a structural description of knowledge, in terms of relations among a set of concepts and a problem solving structure. This description is more adequate to ITS than to LMS based e-learning systems, because of its linkage to a knowledge domain (the set of concepts). For that matter, we've modified the definition, as we'll see next, but keep following a similar qualitative approach.

3.2.4 The SAS Student Model

For the purpose of this work, a SM is a structural description of the learning behavior of a student, in terms of the relations of a set of attributes and metrics that have been defined as useful to snapshot the learning state of the student. This way, we can have a SM that organizes the system knowledge about a student in terms of social background, former academic performance and current behavior. Next figure presents an example of such a SM. In the figure, the notation used will be explained in **3.3** bellow.

$$
SM(paulo)\begin{cases}
submission\_level(shoe\_\mathrm{mod}elling, average)\\
posting\_level(shoe\_\mathrm{mod}elling, low)\\
approval\_level(shoe\_\mathrm{mod}elling, average)\\
had\_attended\_prec(yes)\\
attitude(high)
\end{cases}
$$

Figure 3-2 A qualitative SM

One can argue that this is a fairly simple SM. But a SM doesn't have to be very complex to be useful. The effort to improve accuracy has to be compared to the benefits of the improvement. As Self ([56]) states, "the computational effort to improve accuracy may not justify the extra pedagogical leverage obtained. Computational utility, not cognitive fidelity is the measure for student models".

So, a user model can never be completely accurate; it is usually a rough approximation. It's, so to speak, another instance of the philosophical question that argues that One can never know the Other completely. It is a fact of life. Far from epistemological issues, the user, or student, model is limited by the quantity and quality of the information that can be extracted from the underlying data infrastructures (fourth level of the e-learning systems' architecture above). A student model is also tightly coupled with an e-learning environment, as we've already stressed. When we are dealing with ITSs, we can have SMs more linked to the knowledge domain; in these systems the models try to express the knowledge of the student. If we are using domain independent LMSs, we must design student models separated from the learning matters and more focused on learning behaviors; in these cases, it is harder to be sure of what the student knows. In short, student models have always to be considered in the context they will be used and the goals of the e-learning application.

In the **Figure 3-2, *submission_level(Course, Level)***, ***posting_level(Course, Level)*** and ***approval_level(Course, Level)***, as well as school ***attitude(X)*** are behavioral predicates; ***had_attended_prec(X)*** is a predicate about former academic experience, in this case if the student had attended some prerequisite course.

Student modeling process requires techniques to gather relevant information about the students, to construct the student model. On **Chapter 5** some techniques for acquiring data and initialize the models will be outlined. Student modeling also needs a way to represent the knowledge that is intended to be expressed by the resulting SM. And we should have a means to evaluate the reliability of the information in the SM. These issues we'll discuss next.

MOODLE does not have a true SM. It only has a student profile. However, MOODLE does collect metrics about all sorts of actions made by the users. SMs can be built from the history logs of the platform and updated with student activity logs, as we've seen and shall deepen later. In fact, there is a great amount of discussion about the feasibility of SMs from student interactions with learning platforms. Arroyo et al.[46], Jonsson et al. [47] and Mislevy et al. [48], just to name a few, have some work done on this subject, mainly through the use of Bayesian Networks [49], in conjunction with Data Mining, to model students' behavior.

3.3 Knowledge Representation

Knowledge representation is a crucial factor regarding the success of the operation of a DSS [50], [51], [52], [53]. In the context of a SAS, one must be able to express the facts known, to some degree, about the student in the SM. At the same time, one must be able to figure out the reliability of the facts expressed.

3.3.1 Knowledge Representation for QoI Assessment

A suitable representation of incomplete information and uncertainty is needed, one that supports non-monotonic reasoning.

In a classical logical theory, the proof of a theorem results in a ***true*** or ***false*** truth value, or in an ***unknown*** value. On contrary, in a Logic Program (LP), the answer to a question is only ***true*** or ***false***. This is a consequence of the limitations of the knowledge representation in a LP, because explicit representation of negative information is not allowed. Additionally, the operational semantics applies the Closed-World Assumption (CWA) [54] to all predicates. Usually, LP represents implicitly negative information assuming the application of reasoning according to the CWA.

An extended logic program (**Figure 3-3**), on the other hand, is a finite collection of rules of the form ([50], [57]):

$$q \leftarrow p_1 \wedge \ldots \wedge p_m \wedge \text{not } p_{m+1} \wedge \ldots \wedge \text{not } p_{m+n} \quad (1)$$
$$?\, p_1 \wedge \ldots \wedge p_m \wedge \text{not } p_{m+1} \wedge \ldots \wedge \text{not } p_{m+n} \quad (2)$$

Figure 3-3 An Extended Logic Program

where ***?*** is a domain atom denoting falsity, ***p_i*** and ***q*** are classical ground literals, i.e., either positive atoms or atoms preceded by the classical negation sign ¬. Every program is associated with a set of abducibles. Abducibles can be seen as hypotheses that provide possible solutions or explanations for given queries, here in the form of exceptions to the extensions of the predicates that make the program.

The objective is to provide expressive power for representing explicitly negative information, as well as directly describe the CWA for some predicates, also known as predicate circumscription [55]. Three types of answers to a given question are then possible: ***true***, ***false*** and ***unknown*** and the representation of null values is scoped by Extended Logic Programming (ELP) [52]

We will consider two types of null values: the first will allow for the representation of unknown values, not necessarily from a given set of values; and the second will represent unknown values from a given set of possible values. To see how null values can be used to represent unknown information, let us consider the extensions of some predicates whose attributes resemble that of a SM, namely:

```
had_attended: Student x StrValue

motivation: Student x Value

grade_PA: Student x Value
```

where the former argument denotes a given student and the second represents the value of a particular asset (e.g., ***motivation (ana, 1)*** means that the motivation of the student Ana has the logical value 1).

In **Figure 3-4**, the symbol ¬ denotes strong negation, symbolizing what should be interpreted as false, and the term ***not*** designates negation-by-failure.

$$\begin{array}{l} \text{motivation }(ana, 1) \\ \neg\text{motivation }(S, V) \leftarrow \text{not motivation }(S, V) \end{array}$$

Figure 3-4 An extension of the predicate that denotes the motivation of student Ana

Let us admit that the motivation of another student, say, Diana, has not yet been established. This will be denoted by a null value of the type ***unknown***, as given in the program of **Figure 3-5**: the student has some motivation but it is not possible to be certain about its truth value. In the first clause the symbol $\perp$ represents a null value of an undefined type. It is a representation that assumes any value as a viable solution, but without being given a clue about which value one is speaking about. It is not possible to compute the value of the motivation of student Diana. The third clause of the program (the closure of predicate ***motivation***) discards the possibility of being assumed as ***false*** any question on the specific value of motivation for Diana.

$$\begin{array}{l} \text{motivation }(diana, \perp) \\ \neg\text{motivation }(S, V) \leftarrow \text{not motivation }(S, V), \\ \qquad\qquad\qquad\qquad \text{not exception(motivation }(S, V)) \\ \text{exception(motivation }(S, V)) \leftarrow \text{motivation }(S, \perp) \end{array}$$

Figure 3-5 Motivation of student Diana, with an unknown value

Let us now consider the case in which the value of the motivation for a certain student is foreseen to be 0.60, with an error margin of 0.15. It is not possible to be positive, concerning the motivation truth value. However, it is false that the student has a motivation value of 0.80 or 1. This example suggests that the lack of knowledge may only be associated to an enumerated set of possible known values. As a different case, let us consider the motivation of the student Paulo, that is unknown, but one knows that it is specifically 0.30 or 0.50 (**Figure 3-6**).

$$\begin{aligned}
&\neg \text{motivation } (S, V) \leftarrow \text{not motivation } (S, V), \\
&\qquad\qquad\qquad\qquad \text{not exception(motivation } (S, V)) \\
&\text{exception(motivation } (S, V)) \leftarrow \text{motivation } (S, \perp) \\
&\text{motivation } (\text{ana}, 1) \\
&\text{motivation } (\text{diana}, \perp) \\
&\text{exception(motivation } (\text{carlos}, V)) \leftarrow V \geq 0.45 \wedge V \geq 0.75 \\
&\text{exception(motivation } (\text{paulo}, 0.30)) \\
&\text{exception(motivation } (\text{paulo}, 0.50))
\end{aligned}$$

Figure 3-6 A logical illustration of the motivations for students Ana, Diana, Carlos and Paulo

Using ELP, as the logic programming language, a procedure given in terms of the extension of a predicate called ***demo,*** is given by the program in **Figure 3-7**. This predicate allows one to reason about the body of knowledge presented in a particular domain, set on the formalism referred above. Given a question, it returns a solution based on a set of assumptions. This meta-predicate is defined as:

Demo: Question x Answer

where ***Question*** denotes a theorem to be proved and ***Answer*** denotes a truth value: ***true*** (**T**), ***false*** (**F**) or ***unknown*** (**U**).

$$\begin{aligned}
&\text{demo}(Q, T) \leftarrow Q \\
&\text{demo}(Q, F) \leftarrow \neg Q \\
&\text{demo}(Q, U) \leftarrow \text{not } Q \wedge \text{not } \neg Q
\end{aligned}$$

Figure 3-7 An extension of the meta-predicate *demo*

3.3.2 QoI of Student Models

We have just seen that one may not always be able to reason based only on LP representations of beliefs. We have also seen how to use ELP to express uncertainty and overcome this limitation. In any decision making process, the decision is made without having all the information pertaining to the problem. When the decision maker has to, she makes the decision using the available information, to the best of her knowledge. How much a teacher relies on the diagnostics at hand? How can SM provide her with a measure of the quality of that information?

Let ***i ($i \in 1 \ldots m$)*** represent the predicates whose extensions make an extended logic program that models the universe of discourse; and ***j ($j \in 1 \ldots n$)*** the attributes of those predicates. Let ***$x_j \in [min_j, max_j]$*** be a value for attribute ***j***. To each predicate is also associated a scoring function ***$V_{ij}[min_j, max_j] \rightarrow 0 \ldots 1$***, which gives the score that predicate i assigns to a value of attribute ***j*** in the range of its acceptable values, i.e., its domain (for simplicity, scores are kept in the interval [0 ... 1]), here given in the form:

```
all(attribute_exception_list, sub_expression, invariants)
```

This denotes that ***sub_expression*** should hold for each combination of the exceptions of the extensions of the predicates that represent the attributes in the ***attribute_exception_list*** and the ***invariants***.

This is further translated by introducing three new predicates. The first predicate creates a list of all possible exception combinations (pairs, triples, ..., n-tuples) as a list of sets determined by the domain size (and the invariants). The second predicate recourses through this list and makes a call to the third predicate for each exception combination. The third predicate denotes ***sub_expression***, giving, for each predicate, the respective score function. The QoI with respect to a generic predicate ***P*** is therefore given by ***$QoI_P = 1/Card$***, where ***Card*** denotes the cardinality of the exception set for ***P***, if the exception set is not disjoint. If the exception set is disjoint, the quality of information is given by:

$$QI_P = \frac{1}{C_1^{Card} + \cdots + C_{Card}^{Card}}$$

where C_{Card}^{Card} is a card-combination subset, with ***Card*** elements.

The next element of the model to be considered is the relative importance that a predicate assigns to each of its attributes under observation: $\boldsymbol{w_{ij}}$ stands for the relevance of attribute $\boldsymbol{j}$ for predicate $\boldsymbol{i}$. It is also assumed that the weights of all predicates are normalized, i.e.:

$$\forall i \sum_{j=1}^{n} w_{ij} = 1$$

It is now possible to define a predicate's scoring function, i.e., for a value $\boldsymbol{x = (x_1 \ldots x_n)}$ in the multi-dimensional space defined by the attributes domains, which is given in the form:

$$V_i(x) = \sum_{j=1}^{n} w_{ij} * V_{ij}(x_j)$$

And it is possible to measure the QoI that occurs as a result of a logic program that makes a SM, by posting the $\boldsymbol{V_i(x)}$ values into a multi-dimensional space and projecting it onto a two-dimensional one.

Using this procedure, it is defined a circle, as the one given in **Figure 3-8**.

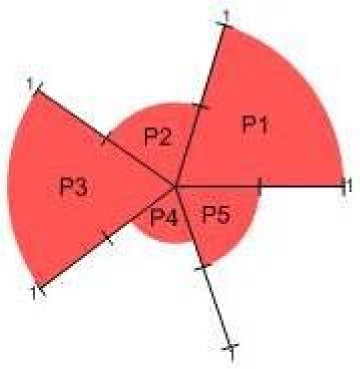


Figure 3-8 A measure of the QoI for a Logic Program or Theory P

Here, the dashed n-slices of the circle (in this example built on the extensions of five predicates, named as $p_1 \ldots p_5$) denote the QoI that is associated with each of the predicate extensions that make the logic program. Now, we can evaluate the QoI of the SMs of Ana and Diana. Let us consider the LP in Figure 3-9 and **Figure 3-10**, which represent a set of beliefs about students, as well as exceptions to those beliefs.

$$
\begin{array}{l}
motivation\ (ana, 1) \\
had\_attended(ana,\ geometry) \\
\neg grade\_PA(S, V) \leftarrow not\ grade\_PA(S, V), \\
\qquad\qquad\qquad not\ exception(grade\_PA(S, V)) \\
exception(grade\_PA(ana, 14)) \\
exception(grade\_PA(ana, 16))
\end{array}
$$

Figure 3-9 An example of the Universe of Discourse for Ana SM

$$
\begin{array}{l}
\neg motivation\ (S, V) \leftarrow not\ motivation\ (S, V), \\
\qquad\qquad not\ exception(motivation\ (S, V)) \\
exception(motivation\ (S, V)) \leftarrow motivation\ (S, \perp) \\
\neg had\_attended\ (S, V) \leftarrow not\ had\_attended(S, V), \\
\qquad\qquad\qquad not\ exception(had\_attended(S, V)) \\
exception(had\_attended\ (S, V)) \leftarrow had\_attended(S, \perp) \\
\neg grade\_PA(S, V) \leftarrow not\ grade\_PA\ (S, V), \\
\qquad\qquad\qquad not\ exception(grade\_PA\ (S, V)) \\
motivation\ (diana,\ \perp) \\
had\_attended(diana,\ \perp) \\
exception(grade\_PA(diana,\ 8)) \\
exception(grade\_PA(diana,\ 11))
\end{array}
$$

Figure 3-10 An example of the Universe of Discourse for Diana SM

QoI associated with students Ana and Diana is depicted in Figure 3-11 and **Figure 3-12**, respectively.

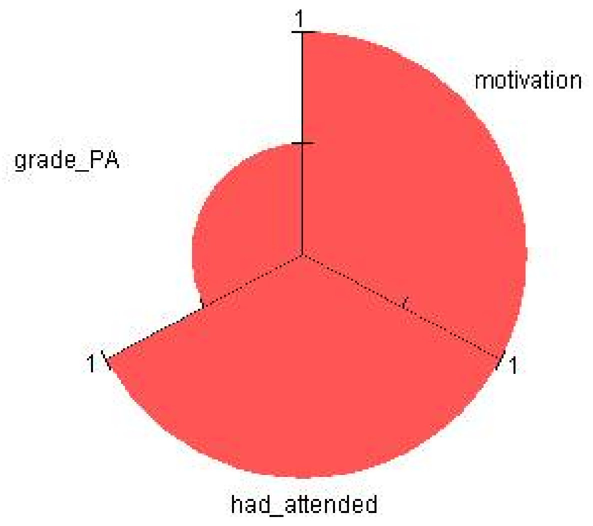


Figure 3-11 A measure of the Quality-of-Information of Ana SM

In order to find the relationships among the extensions of these predicates, we evaluate the relevance of the QoI, given in the form $V_{motivation}$(ana)= 1; V_{grade_PA}(ana)= 0.5; $V_{had\_attended}$(ana)= 1. Roughly, this means that we are sure about the motivation and attendance information of Ana; but we are not so sure about the information on the percentage of right answers (***grade_PA***). As for Diana, we are not sure whatsoever about her motivation and attendance, although we have some assurance (0.5) on her percentage of right answers.

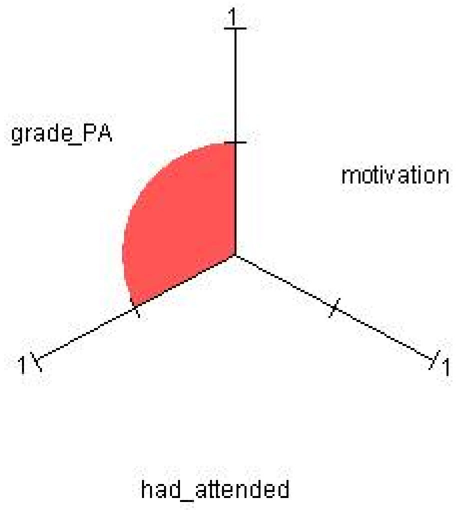


Figure 3-12 A measure of the QoI of Diana SM

3.4 The Basic Framework - Beliefs

The theoretical foundation and basic framework of this project is based on the work of Self [56], who represents a student model based on beliefs about the knowledge that the system and the student have. We'll follow Self's definition but We'll use ELP [57] to express those beliefs. This way, beliefs are represented as predicates. A belief can be assessed as having some degree of truth, as we'll see. This framework relates two sets of beliefs: the student and the computer system set of beliefs.

In this framework, the SM is defined as the representation of the set of beliefs that a system has about a student. The knowledge base, B_c, is defined as the representation of the set of beliefs that a computer system possesses concerning learning behavior. Stated this way, a SM is the subset of the system's beliefs which are beliefs about the student.

On the other hand, the students also possess beliefs about their behavior, their preferences, their performance, etc. This set of beliefs, let's call it B_s, is not known by the system, although it can be guessed and modeled in a SM. Therefore, all the reasoning about the student must be conducted on the basis of the SM. The basic framework, i.e., the relation between these sets is shown in **Figure 3-13**.

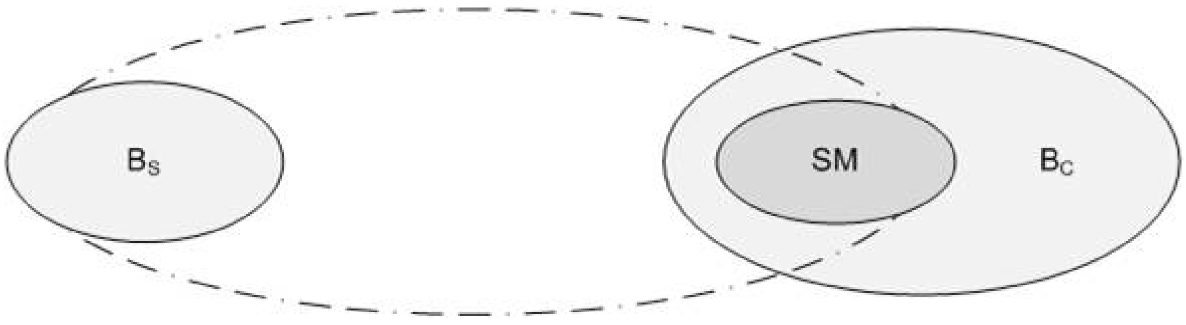


Figure 3-13 The basic framework

In terms of predicate logic, let $B_A p$ denote that a program A subscribes the substance (i.e., the essence of the extension) of predicate p. The belief set B_A configures the extensions of the set of predicates assumed by program A: $B_A = \{p \mid B_A p\}$. By applying this line of thought, we can denote:

B_S as the student set of beliefs;

B_c as the computer system set of beliefs. This set includes the extensions of those predicates that the system believes with reference to the general student behavior;

and

SM, the model of student S, SM(S), is represented as

$$B_C(S)= \{p \mid B_C p(S)\} \qquad \textbf{(1)}$$

meaning that SM(S) is the set of predicates p in B_c about student S.

3.5 Beliefs representation and Storage

The beliefs of the system, B_c, as well as the SM, are represented throughout this work in terms of ELP predicates, as we've said before.

3.5.1 Representation of SMs

The SM is composed of predicates under the form of

$$\text{attribute(value1, ..., valueN)} \qquad \textbf{(2)}$$

We have already seen a SM built up from a set of predicates (**Figure 3-2**). Next figure shows another example of the predicates of a SM, now with three attributes expressed: ***submission_level***, ***posting_level*** and family socioeconomic index, ***family_SEI***.

$$SM\,(paulo)\begin{cases} \ldots \\ submission\_level(shoe\_\bmod elling, high) \\ posting\_level(shoe\_\bmod elling, high) \\ \ldots \\ family\_SEI(average) \\ \ldots \end{cases}$$

Figure 3-14 Representation of the SM of student Paulo

3.5.2 Representation of B$_c$

The core of the knowledge base B$_c$, is composed of ELP predicates that represent beliefs (or rules, if we take the expert systems' point of view), which relate predicates of the SM on the form of implications, C $\rightarrow$ A (**Figure 3-15**).

The left hand side, C, is a collection of conditions that must be met for the belief to be activated (or for the rule to be executed). The right hand side, A, contains the actions to be taken if the conditions in C are met.

$$B_C \begin{cases} submission\_level(X,low), approval\_level(X,low) \rightarrow grade(X,no). \\ approval\_level(X,low), attitude(low) \rightarrow grade(X,no). \\ approval\_level(X,high), attitude(average) \rightarrow grade(X,yes). \\ submission\_level(X,low), attitude(low) \rightarrow grade(X,no). \\ submission\_level(X,average), approval\_level(X,high) \rightarrow grade(X,yes). \\ \ldots \end{cases}$$

Figure 3-15 Beliefs (or rules) representation in B$_c$

In our example, the first rule is saying that if a student has low submission level of work and low approval level on a course, she will (probably) not grade on that course. The third rule states that if the approval level is high and the school attitude is average, the student (usually) grades on the course. The other rules read the same way. Note that, although not expressed, these rules are probabilistic, as mentioned before and will be seen later.

3.5.3 Beliefs storage

In addition to the issue of representation, there is also the issue of storing the knowledge. As far as the storage infrastructure is concerned, we use a relational DBMS to store both the SMs and the knowledge base B$_c$. The SM is stored in the database in connection with the student profile. In fact, the student ID is usually one of the key columns of the tables that contain attributes of the SM. B$_c$ is stored on a table created for that purpose, where the rules can be added, edited and removed. We'll return to this later (4.2.1 Relation between MOODLE Database Columns and Predicates of SM).

3.6 Conclusion

A student model is a computational representation of a student in terms of her biographical, cognitive, behavioral characteristics. Student modeling and SMs depend on the e-learning platform and on the objectives and contexts of its usage. In this work a qualitative SM is used to model the learning behavior of the students, based on data from MOODLE database.

The knowledge of the SAS has the form of ELP predicates, including the representation of incomplete information and uncertainty, suitable to assess the QoI of the SMs.

The theoretical framework of SAS follows the cited work of Self [56] and its concepts of student set of beliefs, B_s, computer system set of beliefs, B_c, and student model, SM. The knowledge base B_c is comprised of a set beliefs (rules) on the form of implications, $C \rightarrow A$, that relate predicates of the SM. Whenever the condition predicate(s), C, occurs, the action predicate(s), A, are triggered.

4 The Student Assessment System

In this chapter the SAS architecture is depicted, starting with the review of the technological infrastructure that underlies the system and then presenting its components, the Student Model Builder and the Student Model Diagnoser. After that, it is explained how the module Student Model Builder operates, including the relation between MOODLE database columns and attributes of the predicates that compose the SM. Finally, this chapter also describes the Student Model Diagnoser, including its inference model, the representation of its beliefs (or rules), and its implementation.

We use the terms "belief" and "rule" almost as synonyms. It must be said that, in the context of this work, "belief" is the correct term to name the probabilistic nature of the predicates. On the other hand, "rule" is the correct term to use when we talk about inferential expert systems. So, depending on the setting, a belief becomes a rule.

4.1 System Architecture

A SAS configures a DSS made of two modules (**Figure 4-1**): a module to create and update the SM; and a module to diagnose the learning path of the student. The former is the Student Model Builder (SMB); the latter is the Student Model Diagnoser (SMD), as we've just stated.

4.1.1 Technological choices

Systems' design especially, that of commercial systems, is limited by the technological and human resources available, by the budget for the project and by the time to market. In the case of this project, although not commercial, there were similar constraints. In fact we could not acquire any software or hardware; as we had no budget at all, we were "restricted" to the infrastructure of CFPIC and to free or Open Source tools and tools provided by the University of Minho (UM) in the scope of the Master Class. We also had a deadline we must observe. As for the human resources, we won't talk about them: they are at stake here.

With this in mind, becomes clear that the technological choices suffered from some limitations. But, hey, that's what engineering is all about, isn't it?

The SAS, as we've already seen, is a system that gathers information from an e-learning platform and several other data sources to evolve models of the students and produce diagnosis of their learning careers.

The e-learning platform chosen was MOODLE because it is Open Source software (therefore it is free) and it was the e-learning platform used in CFPIC Professional School. The choice of MOODLE shaped the choice of the development language, PHP, and also that of the database engine, MySQL DBMS [38], because these are technologies closely tied to MOODLE. In fact, MySQL is, by default, the underlying DBMS of MOODLE platform, and MOODLE is written in PHP and accepts PHP plug-ins. SAS is intended to be installed as a MOODLE PHP plug-in.

In addition, CFPIC Professional School operates MOODLE on top of Microsoft Windows Server [58] operating system. This is a well known network operating system that supplies the services needed for SAS to communicate, namely TCP/IP protocol.

Finally, the ELP programs that make up the SMs and B_c, as well as the diagnosis inference engine, were developed using the SICStus Prolog [62] programming platform, provided by the Informatics Department of the University of Minho to be used in the context of the MSc.

4.1.2 Architecture of the system

The architecture of SAS is depicted on **Figure 4-1**, below. We can see the main components of the system: the SMB, the SMD, the MOODLE e-learning system, the knowledge base B_c and the student models SM.

The SMB is the subsystem responsible for initializing and evolving the SMs. It is sensible to changes in MOODLE database, for every student's interaction, and uses it to trigger updates to the SMs.

As for the SMD, as we'll see in detail later on, this module picks the SM and computes its QoI. If the QoI is above a certain threshold, SMD progresses to confront the knowledge about a student, represented in her SM (a set of predicates), with the beliefs of the system, B_c (a set of predicate implications), to produce a diagnosis. If the QoI is below that threshold, the SMD asks for the missing information. A set of diagnosis of the SMs of S comprises the follow up of student S learning path, in the sense that it represents her evolution, or involution, during the learning process.

The diagnosis is the output of the system and it is delivered to teachers and other school personnel.

The external data sources are foreign data storage infrastructures, or information provided by the students themselves, on the form of answers to questionnaires, that complement the knowledge of the system about the student.

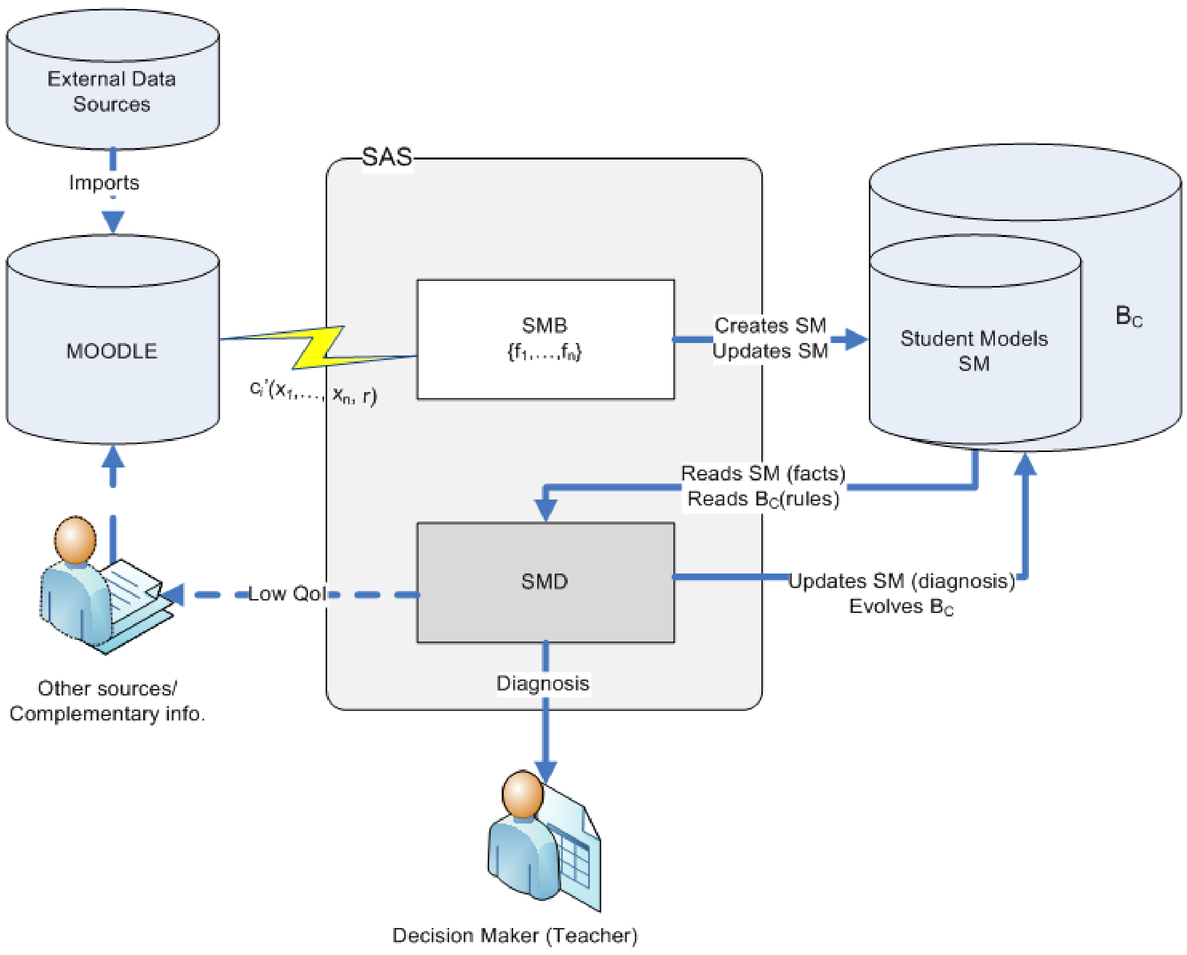


Figure 4-1 SAS architecture

4.2 The Student Model Builder

SMB is a component of the SAS that resides and operates within MOODLE platform. It gathers information from several sources to initialize and evolve the SMs. Before presenting the operation of SMB, and to ease its understanding, we'll start by explaining the relation between MOODLE database columns and predicates in the SM.

4.2.1 Relation between MOODLE Database Columns and Predicates of SM

As we've seen, the SM is composed of a set of predicates of the type defined in (**2**), made up of static and dynamic attributes that were considered (by teachers, psychologists, etc.) relevant to the diagnosing process.

There is a connection between attributes of predicates in SMs and columns in MOODLE database. Generally, if there is an attribute useful to build relevant predicates for the production of diagnosis, there should be a column, or a set of columns, in MOODLE database, to model that attribute. This is true for both static and dynamic attributes. There's no need to say that relevant predicates should always be present in the SM.

For instance, a column to identify the course already exists in MOODLE database (***mdl_course.id***); but columns for attributes like the level of posting of a student on a course forum, for instance don't. If one feels that these attributes are important to model the student (to evaluate the student engagement on the course, for instance), one must add columns to the MOODLE schema to represent them. Next figure exemplifies an extension to MOODLE schema to include the attribute *posting_level*.

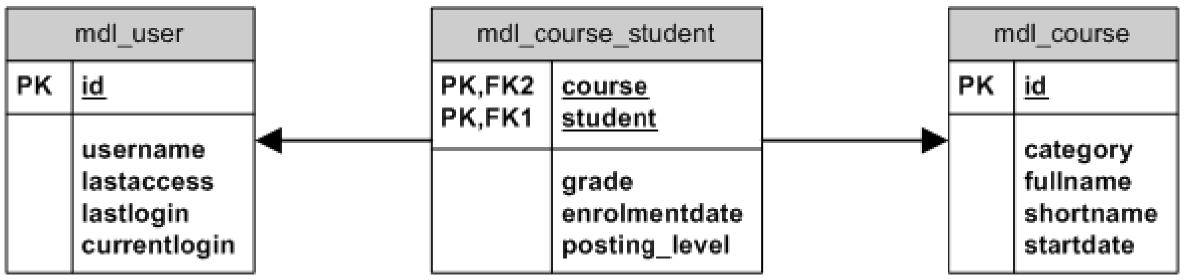


Figure 4-2 Extension to MOODLE schema showing the attribute column *posting_level*

Generally, let *SM(s)* be the SM of student *s* comprised of a set of predicates modelling the learning behaviour of *s*. Let *C*, be a set of columns belonging to a set of MOODLE tables which somehow are involved in the student modelling. And let *T* be a table that has a subset *C'* of columns which were considered relevant for the diagnosing process. This table is also composed of a set of key columns, *K*. In these conditions we have

$$\begin{aligned}
&C' = \{c_1',\dots,c_n'\} \subset C \qquad\qquad (3)\\
&K = \{k_1,\dots,k_m\}\\
&C' \cap K = \{\ \}\\
&c_i'(x_1,\dots,x_m,r) \in SM(s), x_j = v(k_j), r = f(k_i,\dots,c_i)
\end{aligned}$$

The columns in *C'* (as well as those in *C*) can be original MOODLE columns or columns added latter to the MOODLE schema. The columns c_i' for student *s* "become" attributes of SM predicates with arguments $x_{ij} = v(k_j)$, i.e., the values of the set of key columns of the table *T* for student *s*. Usually there is some sort of relation between attribute columns in *C'* and the argument columns in *K*. This relation $f(k_i,...,c_i)$ may be of many kinds, ranging from simple equations to complex mathematical or logical functions. The argument *r* of the attribute is the solution of the function $f(k_i,...,c_i)$, being c_i a column of MOODLE database that the attribute c_i' depends on. $f(k_i,...,c_i)$ is implemented as a function stored in the database (stored procedure, user-defined function), as we'll see.

As an example, to represent the posting level a student has on a course, we have

- A set *C = {student, course, sum_of_posts_in_foruns, posting_level, ...}* of columns from MOODLE database;
- A subset *C' = {posting_level}* of attribute columns;
- A subset *K = {student, course}* of key columns;
- Values *v(student) = s*, *v(course) = Shoe_Modelling*;
- A predicate *posting_level(course, level)*;
- A function ***f(****course, sum_of_posts_in_foruns)=level*, as such: *"if course = Shoe_Modelling and sum_of_posts_in_foruns >= 10 then level = AVERAGE"*;

The set of predicates *SM(s)* includes {..., *posting_level(Shoe_Modelling, AVERAGE)*, ...}. Some predicates may not be defined for some SMs. This occurs when we don't have the adequate QoI about a number of attributes needed to build some predicates. This can happen because some, or a combination of, the following situations can occur

- We don't have data (*x=v(k)*) for some of the columns *k* that compose the attributes *c'* of the predicates of the SM;
- We don't have values for c_i;
- We don't know the function $f(k_i,...,c_i)$.

For instance, *posting_level*, although defined for student *s* above, may not be so for student *t*, because either we don't have information about posts for the course *t* is enrolled in (say, Shoe_Sewing), or we don't have a relation $f(k_i,\ldots,c_i)$ that links *posting_level* and Shoe_Sewing. In this situation, *posting_level* has a null value for student *t* on course Shoe_Sewing.

As you have noticed, we have suppressed the value *s* for student in the predicate *posting_level(Shoe_Modelling, AVERAGE)*. This is because the predicate belongs to the SM of *s* which constitutes the universe of facts.

4.2.2 The SMB operation

This module, as said before, is sensitive to changes in MOODLE database. SMB is called to update the SMs whenever relevant information in MOODLE database gets modified. This is achieved by using a trigger mechanism: every time a column, or columns, belonging to *C*, referred in (**3**), suffers a change, SMB is fired to alter the value of *r* and, therefore the predicate $c_i'(x_1,\ldots,x_m, r)$ of the SM. There could be several ways of carrying this idea out. My implementation consists on declaring triggers on tables that possess $c_i \in C$ columns. For instance, one can declare a trigger on ***mdl_forum_posts*** (or on ***mdl_log***) to call SMB whenever the student *s* posts a message on Shoe Modeling course forum (**Figure 4-3**). SMB would call the stored procedure that implements the function *level=f(course, sum_of_posts_in_foruns)*, to compute the new value of *level* and update the corresponding *posting_level* column as well as *posting_level* predicate in *SM(s)*. The SMB is implemented as a set of stored procedures or User Defined Functions (UDF) within MOODLE database.

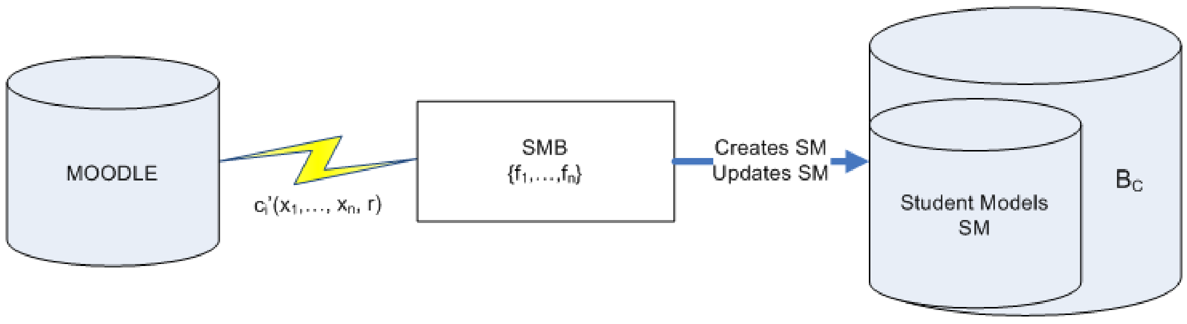


Figure 4-3 Operation of the SMB

The SMB is indeed the responsible for initializing and evolving the SM based on changes on the student's data. This data is synthesised on a set of columns which model attributes on the SM, as we've seen on **4.2.1**.

4.3 The Student Model Diagnoser

There are, in fact, two sources of information on the basis of which a SM may be updated: the student's inputs to MOODLE, and the current contents of B_c. Generally, the SMD picks up the SM of a given student and evaluates the SM's QoI; if the QoI meets the requirements defined[5], the SMD starts its inference process of examining the SM based on the rules in B_c, in order to produce a diagnosis. Finally, it updates the SM and outputs the new version of the SM, which includes the diagnosis. This diagnosis results from evaluating the system set of beliefs, B_c, for a particular student, and is expressed in terms of grading or not grading on a given course. **Figure 4-4,** below, stands for the SMD structure.

SMD is an expert system on this particular domain of SM diagnosing. As an expert system, it has a knowledge base, a problem solving function and a user interaction function. The SMD expert system is based on a model well established in the literature [59], where the data is separated from the inference engine and the user interface; these two functions compose one module, called the expert system shell, or simply the shell. By separating the data from the shell, we are separating the domain knowledge from the reasoning process, seeking, that way, generality and simplicity.

The core of SMD is its inference engine. Before presenting the inference engine, we'll explain the inference model and the representation of the rules (or beliefs) that make up the knowledge base B_c.

[5] The definition of acceptable QoI is the responsibility of the decision makers and is out of scope of this work.

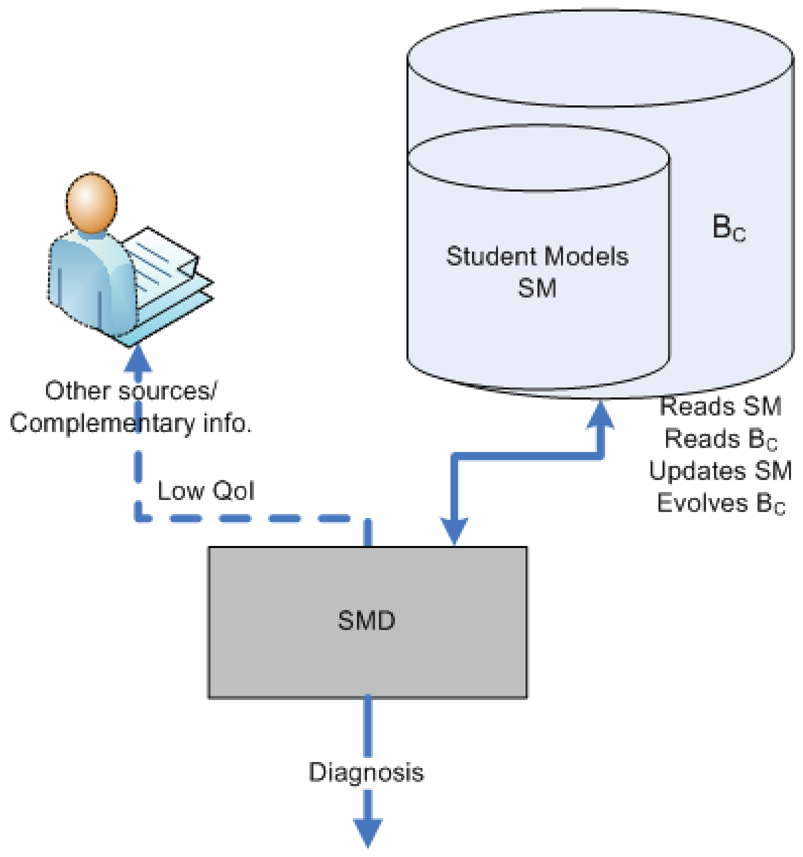


Figure 4-4 Student Model Diagnoser

4.3.1 The Student Model Diagnoser Inference Model

The SMD uses a forward chaining [60], or data driven, inference model similar to that of pattern programming systems [59] to produce a diagnosis. We feel that this inference model is better suited to our problem than backward chaining [60], because we are not trying to prove a theory, i.e., we are not searching from goal to data, but the other way around: we are searching from data to goal, i.e., we are trying to devise some meaning from the available data (going from the particular to the general). Some ideas for the inference engine, particularly some code snippets, were also taken from [61].

SMD is composed of the following three components:

- The rule set knowledge base (B_c) and the facts under observation (SM);
- The inference engine;

- The user interface, which gives the user control over the system, and explains how the conclusions were achieved.

The inference engine takes the SM and the rules, does the necessary reasoning and outputs an updated SM with a diagnosis. The user interface has functions to control the operation of SMD, and functions to explain how the results were achieved.

The SMD inference engine and user interface, as well as the knowledge base rules and the SM facts, are coded in SICStus Prolog [62] because when it comes to express knowledge and to reason over that knowledge Prolog allows for a great combination of power, simplicity and flexibility. Specifically, some very useful features of Prolog are:

- Definition of operators (functors) enhances the readability of the rules;
- Built-in backtracking search eases the rule selection;
- Built-in pattern matching (unification) eases comparison of data;
- Built-in database that provides an easy representation of working storage, where SMs are loaded to, as we'll see later.

4.3.2 The representation of rules in B_c

The rules for expert systems are usually written in the form:

- IF condition C THEN action A

This is equivalent to an implication

- $C \rightarrow A$

The representation described in **3.5** follows the second notation and, complemented with some syntax sugar and the probabilistic factor, it is suitable to express the knowledge of the SAS expert system and, thus, to support the reasoning process. This representation of rules is as follows:

ID:[1:Condition1, …, N:ConditionN] ==> [Action1, …, ActionN]:(CF) **(4)**

where

- ID – is a unique identifier of the rule;
- N – is an identification for the condition;
- Condition – is a pattern to match against working storage (SM);
- Action – is an action to take when the Condition matches some fact of the SM;
- CF – is the Certainty Factor, defined as CF = confidence, for a given support of B_c, as will be explained in **Chapter 5 Initializing the Student Assessment System**.

Each condition is a Prolog clause. In general, the condition pattern is matched against those stored in working storage using the unification process.

The actions are:

- assert(X) - adds the term X to working storage;
- retract(all) - removes all of the working storage terms which were matched in the left hand side of the rule being executed;
- retract(N) - retracts left hand side condition number N from working storage;

As an example, **Figure 4-5** shows a knowledge base using the syntax just defined, including the Certainty Factor, CF. This knowledge base is the basis for our examples from here on, and it is derived from the rules extracted from MOODLE database plus some rules added manually, as we'll see with detail in **Chapter 5 Initializing the Student Assessment System**.

$$B_c \begin{cases} \textit{r1:[1:submission\_level(X,low), 2:approval\_level(X,low)]} \Rightarrow \textit{[retract(all), assert(grade(X,no))]:(1.0).} \\ \textit{r2:[1:approval\_level(X,high), 2:attitude(high)]} \Rightarrow \textit{[retract(all),assert(grade(X,yes))]:(1.0).} \\ \textit{r3:[1:approval\_level(X,low), 2:attitude(low)]} \Rightarrow \textit{[retract(all),assert(grade(X,no))]:(1.0).} \\ \textit{r4:[1: approval\_level(X,high),2:attitude(average)]} \Rightarrow \textit{[retract(all),assert(grade(X,yes))]:(0.97).} \\ \textit{r5:[1:submission\_level(X,low),2:attitude(low)]} \Rightarrow \textit{[retract(all),assert(grade(X,no))]:(0.96).} \\ \textit{r6:[1:submission\_level(X,average)2: approval\_level(X,average)]} \Rightarrow \textit{[retract(all),assert(grade(X,yes))]:(0.95).} \\ \textit{r7:[1:submission\_level(X,average)2: approval\_level(X,high)]} \Rightarrow \textit{[retract(all),assert(grade(X,yes))]:(0.95).} \\ \textit{r8:[1:posting\_level(X,low),2: attitude(low)]} \Rightarrow \textit{[retract(all),assert(grade(X,no))]:(0.95).} \\ \textit{r10:[1:approval\_level(X,average),2: attitude(high)]} \Rightarrow \textit{[retract(all),assert(grade(X,yes))]:(0.94).} \\ \textit{r11:[1:approval\_level(X,average)2: attitude(average)]} \Rightarrow \textit{[retract(all),assert(grade(X,yes))]:(0.94).} \\ \textit{r12:[1:submission\_level(X,average)2: posting\_level(X, high)]} \Rightarrow \textit{[retract(all),assert(approval\_level(X,average))]:(0.93).} \end{cases}$$

Figure 4-5 Redefinition of the B_c using the newly defined syntax

Note that *grade(X, yes)* and *grade(X, no)* are the diagnostics that will be expressed in the SM as one of the outcomes of the reasoning process. The other is the version of the SM that results from the assertion and retraction of facts. Whenever we can, we should remove the redundant rules from the knowledge base B_c. This way we simplify the operation and the readability of B_c.

Both B_c and SM are exported from MOODLE database into Prolog readable files. This export process is automatic and may occur whenever we want to investigate a student career (**Figure 4-4** above).

4.3.3 The implementation of the inference engine

We've followed the standard approach of building a shell that includes predicates to manage the user interface and predicates to make up the inference engine. This inference engine is coded in the predicate `run/0` as shown in the next figure.

```
run :- call(ID : Condition ==> Action : (CF)),
       try(ID, Condition, Action),
       update_cf( CF),
       ui([' Rule ',ID,' fired with Certainty Factor : ',CF]), nl,
       !, run.
run : - nl, print_resu lt.
```

Figure 4-6 Simplified view of SMD inference engine

The *ID* argument is the rule *ID* in (**4**); the *Condition*, *Action* and *CF* arguments are also the same as in (**4**). The behaviour of this shell is pretty simple:

- Before starting the inference cycle, the user prepares the system. This preparation includes loading the knowledge base and SM files through the Prolog interpreter and asserting the SM into the working storage.
- Next, the user calls the inference engine, denoted by the predicate `run/0`.
- The inference engine searches the first production rule in B_c that matches the facts in the working storage (SM), executes it and updates the Certainty Factor. Then it repeats the process until no more rules match.
- When this happens, the second clause of `run/0` is executed and the inference ends. The second clause prints out the contents of the working storage (SM, diagnosis and the resulting Certainty Factor), showing what was determined during the run.
- The user is notified on every step of the inference process using the user interface function (*ui(L)*).

The predicative expression *try(ID, Condition, Action)* on **Figure 4-7** makes up the core of the inference engine. If `match/1` fails, `run/0` backtracks to find the next rule. The *Condition* is passed to `execute/2` so retract statements can find the facts to retract, i.e., those facts that compose the *Condition*.

```
try(ID, C, A) :- ui([' Trying  rule id : ', ID]),
         match(C),
         execute(A,  C).
```

Figure 4-7 The **try/3** predicate

The `match/1` predicate recursively goes through the database of conditions, searching for those (in fact, for the first) that match the left side.

```
match([]) : -!.
match([N : First | Rest]) : -!,
        fact(First ),
        match(Rest ).
```

Figure 4-8 The `match/1` predicate

If **match/1** succeeds, then `execute/2` is called. It executes the *Action* list of actions.

```
execute([] , _): -!.
execute([F irst | Rest], Condition) : -
        take(Fir st, Condition) ,!,
        execute(Re st, Condition) .
```

Figure 4-9 The **execute/2** predicate

The `take/2` predicate retracts **all** predicates or predicate **N** of a *Condition*; otherwise it passes control to `take/1` which retracts and asserts a single fact:

```
take(retra ct(N), C) : - (N == all; integer(N) ),
        retr(N, C), !.
take(A, _) : -take(A), !.
take(retra ct(X)) : -retract(fa ct(X)), ui([' Using : ',X]), !.
take(asser t(X)) : - asserta(fa ct(X)), ui([' To conclude : ',X]), !.
```

Figure 4-10 Predicates `take/2` and `take/1`

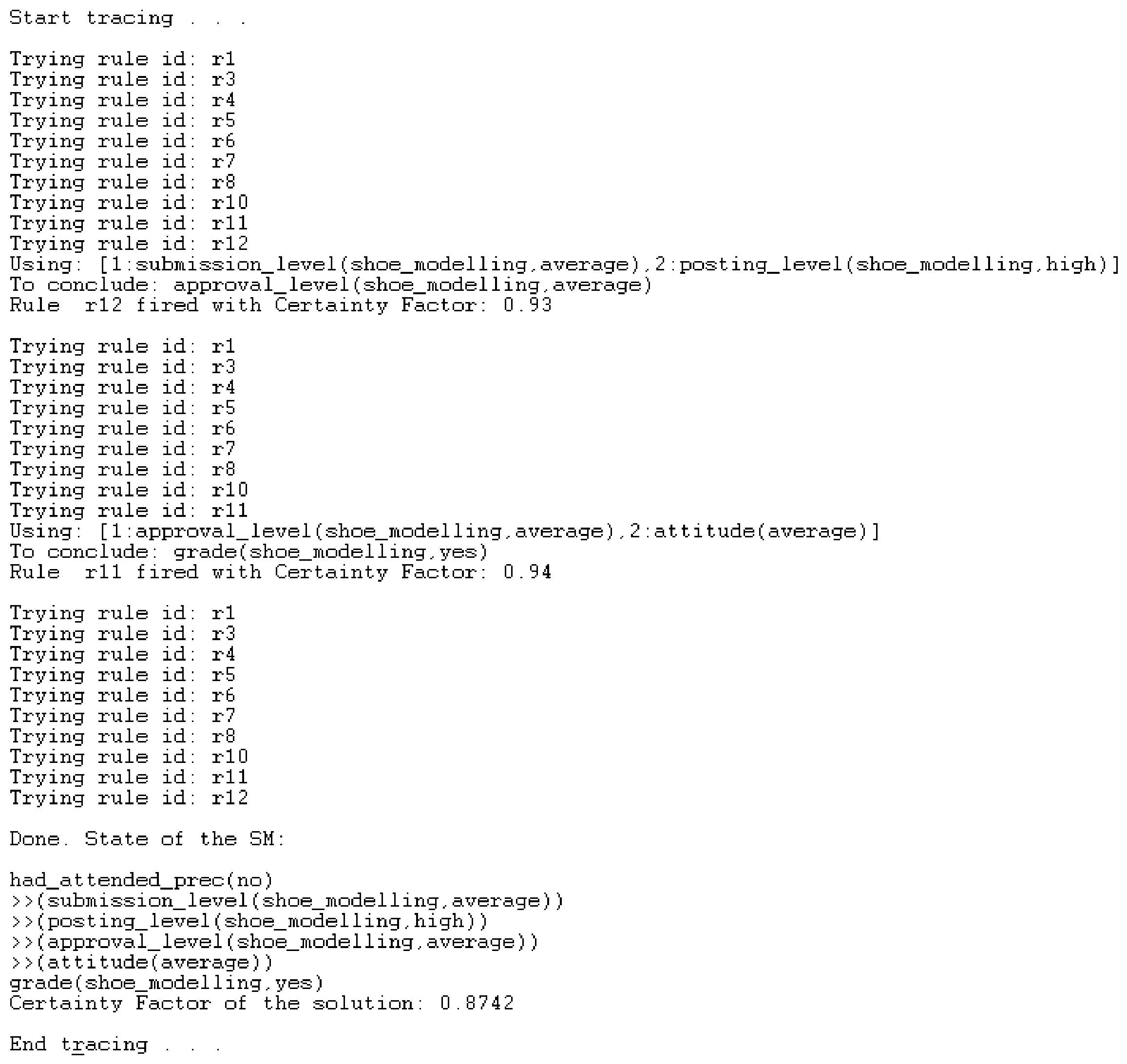
The `retr/2` searches for conditions with the same identification (**N**) and retracts them. If **all** was indicated, then it retracts all of the conditions.

Finally, the Certainty Factor of the solution (diagnosis) is the product of the Certainty Factors of the rules that have matched.

4.3.4 Explanation of the SMD reasoning process

Explanations for forward chaining systems are difficult to implement. This is because each rule modifies the working storage state, thus covering the older state.

Next figure shows a sample trace of the inference process.



```
Start tracing . . .

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Trying rule id: r12
Using: [1:submission_level(shoe_modelling,average),2:posting_level(shoe_modelling,high)]
To conclude: approval_level(shoe_modelling,average)
Rule  r12 fired with Certainty Factor: 0.93

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Using: [1:approval_level(shoe_modelling,average),2:attitude(average)]
To conclude: grade(shoe_modelling,yes)
Rule  r11 fired with Certainty Factor: 0.94

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Trying rule id: r12

Done. State of the SM:

had_attended_prec(no)
>>(submission_level(shoe_modelling,average))
>>(posting_level(shoe_modelling,high))
>>(approval_level(shoe_modelling,average))
>>(attitude(average))
grade(shoe_modelling,yes)
Certainty Factor of the solution: 0.8742

End tracing . . .
```

Figure 4-11 Trace of the inference process

The most useful information in debugging a forward chaining system stems from a trace facility. In SMD, the user interface function logs the operation of the system directly to the screen or to a file for future analysis. Basically, we want to know what rules were fired and the effects they had on the working storage.

We can see that the user interface informs us about the decisions taken, as well as the rules affected (marked **>>**) and, of course, the diagnostic achieved with a given Certainty Factor.

4.4 Conclusion

The architecture of the SAS is composed of two modules, the SMB and the SMD. The former is responsible for building and evolving the SMs. It rests inside MOODLE database. In fact it is an extension to MOODLE database schema. It is composed of a set of triggers that are sensitive to relevant data changes and a set of stored procedures/user defined functions that are called by the triggers to update the SM attributes and predicates.

The second is responsible for producing the diagnosis of the student learning paths. SMD uses a forward chaining inference model of the same kind of pattern programming systems to produce a diagnosis. It is comprised of a rule set knowledge base (B_c), a set of facts (SM), an inference engine and a user interface, which gives the user control over the system, and explains how the diagnosis was achieved.

5 Initializing the Student Assessment System

The initialization of the SAS comprises the initialization of the SMs and the initialization of the knowledge base B_c [63].

This chapter explains the process of initialization of the SMs and B_c. It also describes the role of the SMB and the role of the technicians during this process.

5.1 Initializing the Student Models

The SM can be initialized through import of the student's data from external data sources (biographical and academic databases), through question forms presented to the student, through system default assumptions about the student (socio-economical status, knowledge acquired, preferences and other templates), and through a combination of these methods. Ideally, the initialization of the SM would take place at the time of student's admission at the school.

The attributes that are subject of initialization are the static ones. The values of the dynamic attributes are collected during the learning process.

5.1.1 Initializing the SM with Information from External Data Sources

With this method the values of the attributes that make up the predicates are imported from external data sources into MOODLE database. Typically, socio-economical, cultural, academic history and behavioral indicators are imported, this way updating (or becoming) attributes of the predicates of the SM.

In the event that the source databases are relational, SQL commands can be used to work out the importing process. Being otherwise, one can create intermediate files with formats that are common to both systems (CSV files, XML files, etc.).

During the importing process, for a given student, attributes and values of MOODLE database are being changed. At each of these changes SMB module is notified to create or update predicates on the SM. On **4.2.1**, we've see in detail how this process works. At the end of the importing, the first version of the SM is produced.

5.1.2 Initializing the SM through Forms

With this method, forms with questions about socio-economical and cultural status and questions about former academic history, as well as questions about other indicators considered relevant, are presented to the student. These tests can be very boring for the student and cause unreliable answers. Furthermore, the conception of these tests is complex in itself and should be committed to specialists, not always available.

Once completed, the forms' data must be inserted in MOODLE. Like in the former method, SMB is notified to insert or update a predicate in the SM. When all questions are inserted we'll have the first version of the SM.

5.1.3 Initializing the SM through Default Assumptions

This method assumes that the students can be classified into stereotypes [56], or classes, to which SM templates are associated. If one believes that a certain student belongs to a stereotype *E*, which has a SM template, SM_E, one can initialize the SM of that student with the values of SM_E. For instance, a SM_E for students that have enrolled in Informatics can contain predicates like the ones in the figure bellow:

$$MA_E \begin{cases} \ldots \\ like\_computers(E) \\ \neg math\_grade\_level(E, low) \\ \ldots \end{cases}$$

Figure 5-1 SM_E template for students that had enrolled in Informatics

In this example it is assumed that students that enroll in Informatics like computers (in some sense) and have approval level in mathematics. In fact, it happens that not all students that enroll in Informatics like computers; and there are some that did not have approval level in math.

This method produces, at once, the first version of the SM.

5.1.4 Initializing the SM through a Combination of Methods

The best way to solve the problems inherent to each method just presented is to combine some or all of them. The importing from external databases is the preferred method whenever those databases can provide all the necessary data. If that is not possible, one can import the data supplied and complement with questionnaires and/or default assumptions.

The default assumptions method is adequate when one can define a set of stereotypes that classify a population of students with a high level of accuracy. When we can’t establish the stereotypes, the method becomes inadequate. This method should improve with a clustering analysis of student populations.

The method of question forms appears to be useful only as a complement of the other two.

5.2 Initialization of the Knowledge Base B_c

MOODLE, as other LMSs, offers several usage reports. However, in scenarios where there are many users and a great volume of transactions, it is hard for the teacher, or for the system administrator, to extract information that would allow deriving rules useful to produce diagnosis. One option is to use Data Mining techniques to extract knowledge and produce associative rules on the form of implications (like the ones in **Figure 4-5** above).

Generally, the idea is:

1. Find patterns, correlations and associations, with given thresholds of support and confidence, on MOODLE database;
2. Pick the ones considered acceptable;
3. Insert them in B_c.

The thresholds of support and confidence are fixed by the users. These metrics introduce uncertainty values to the rules discovered, as we've referred about the rules on **Figure 4-5**. The higher the support and confidence values, the more frequent the rules are to occur in the database. To say that the rules in the knowledge base B_c have a certain support is to say that B_c has that support.

The methodology employed for mining association rules is similar to, and based on, that used by Agrawal, Imielinski and Swami [64], and Agrawal and Srikant [65]. For this matter we've used data from past interactions between the students and MOODLE. Next figure depicts the process of initialization of B_c.

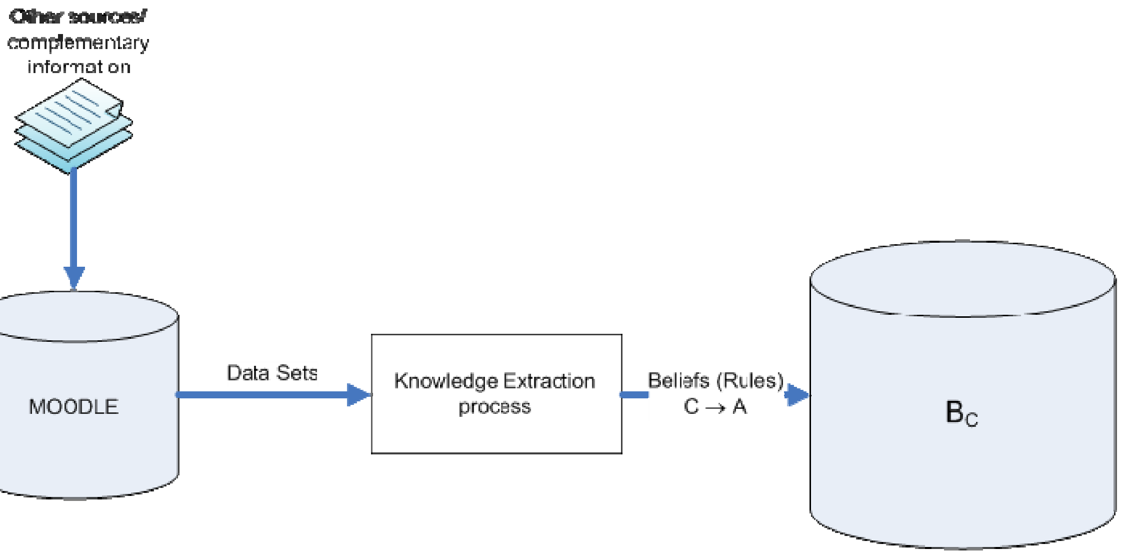

Other sources/ complementary information
MOODLE
Data Sets
Knowledge Extraction process
Beliefs (Rules) C → A
Bc

Figure 5-2 Rule extraction to initialize B_c

The knowledge extraction on LMSs follows the four general steps of any Data Mining process: Data selection, Pre-processing, Data Mining and Results' evaluation.

The Data Mining tool we've used is Weka [66], because it's an Open Source software with all the tools needed to extract the knowledge. Weka has also the advantage of providing an API written in Java that can be called from any application.

The Data Mining process described next intends to analyze the relations between three indicators. In fact, the knowledge base initialization is complete only when all the variables considered relevant to the diagnosing and following of students had been object of association rules extraction. That is, the process that will be described will have to be repeated for all the relevant indicators.

5.2.1 Data selection

It has already been said that all users' interactions with MOODLE are stored in its relational database. It has also been referred that the database is supported, in our implementation, on MySQL engine. MOODLE database schema is composed of more than 100 tables. As an example of Mining to discover relations between indicators to initialize the knowledge base B_c, we propose to analyze, for a given student, enrolled on a given course, the relations between the indicators

- Number of assignments submitted (*submission_level*),
- Number of posts in the forums (*posting_level)*,
- Approval level on the tests (*approval_level*) and
- Grading of the student on that course (*grade*).

For this matter, we need to relate the data in the following tables:

- ***mdl_user***: data (biographic, social, economical and such) about the user
- ***mdl_assignment***: data about the assignments.
- ***mdl_assignment_submissions***: data about the assignments submitted.
- ***mdl_forum_discussions***: data about the forum discussions.
- ***mdl_forum_posts***: data about the posts.
- ***mdl_quiz***: data about the quizzes.
- ***mdl_quiz_grades***: data about quizzes' approval level.

Even though there are tenths of courses with hundreds of enrolled students in the MOODLE CFPIC platform, most teachers use the platform only to produce and deliver home work assignments to their students. Few of them make tests or moderate and stimulate discussions in forums. In these conditions it was necessary to select the courses that did all three activities. These courses are seven with a universe of 76 students enrolled.

For the selection of data several views were created that sum up the information required. For example, to know the number of quizzes made per course the view ***dmv_quizzes_course*** was created:

```
create view `dmv_quizzes_course` as SELECT course, count(id)
as n_coursequizzes FROM `mdl_quiz` group by course
```

To know the tests per course where the students had approved/unapproved level, the view ***dmv_quiz_results*** was created:

```
create view `dmv_quiz_results` as SELECT q.course,qg.userid
as student, qc.n_coursequizzes, sum(if(qg.grade >=
(q.grade/2),1,0)) as approved FROM `mdl_quiz` q join
`mdl_quiz_grades` qg on q.id=qg.quiz inner join
`dmv_quizzes_course` qc on q.course=qc.course group by
q.course,student
```

For assignments and posts, the equivalent views were created:

- ***dmv_assignments_course***: gives the number of assignments per course.
- ***dmv_assignment_results***: gives, per course, the students that submitted assignments.
- ***dmv_forum_posts_course***: gives the number of posts per course.
- ***dmv_forum_posts_results***: gives, per course and per student, the number of posts produced.

5.2.2 Pre-Processing

The Pre-processing step of Data Mining has the objective of preparing the data on the views to take the adequate shape to be exploited by Weka, particularly, by its Apriori algorithm [65] implementation.

In this step we have used MySQL Administrator tools to execute SQL queries that would allow for the production of a Data Set which related assignments, quizzes and posts. SQL code bellow exemplifies a query to relate the three entities:

```
select a.course, a.student, a.n_courseassignments,
a.submited, if(a.submited/a.n_courseassignments=1, 'HIGH',
if(a.submited/a.n_courseassignments<0.33, 'LOW',
'AVERAGE')) as submission_level, q.n_coursequizzes,
q.approved, if(q.approved/q.n_coursequizzes=1, 'HIGH',
if(q.approved/q.n_coursequizzes<0.33, 'LOW', 'AVERAGE')) as
approval_level, p.n_posts,
if(p.n_posts>3,'HIGH',if(p.n_posts<2,'LOW','AVERAGE')) as
posting_level from `dmv_assignment_results` a inner join
`dmv_quiz_results` q on a.course=q.course and
a.student=q.student inner join `dmv_forum_posts_results` p
on a.course=p.course and a.student=p.student where
n_coursequizzes is not null and n_posts is not null order by
a.course,a.student
```

It must be noted that, for each activity (assignments, quizzes, posts), the number of interactions made by each student was discretized in three levels, HIGH, AVERAGE and LOW, as a function of the total of interactions for these activities made per course. The result of this query was exported to a CSV file.

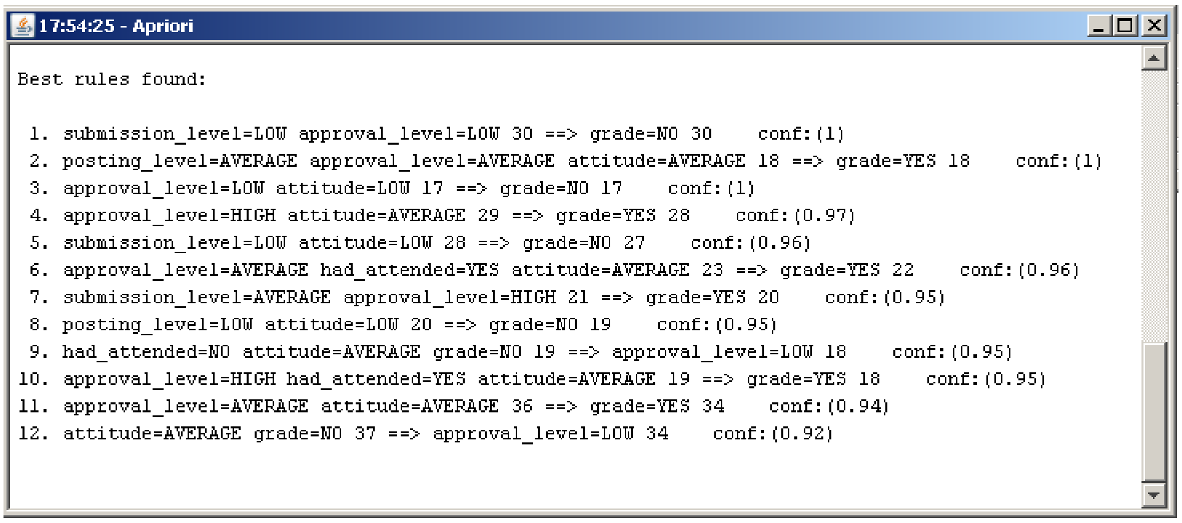
Finally this CSV file was imported to Weka and the Apriori algorithm was applied to extract the association rules.

5.2.3 Apriori Algorithm Application

The Apriori algorithm generates rules on the form of implications $C \rightarrow A$. Each rule has two related metrics, support and confidence. *Support(C)* indicates the percent of transactions (interactions) that contains *C*. Confidence of the rule is defined as: *confidence(C → A) = support(C ∪ A)/support(C).*

As we are only interested on the levels of assignments submission, quizzes approval and posts made, we can remove from the Data Set entities like course and student. By applying Apriori algorithm we derive a set of implication rules from MOODLE records of past student interactions with the selected courses, and establish the beliefs that will comprise the first version of B_c. Besides automatically discovering association rules, these can also be manually configured by teachers, psychologists and others. The problem of figuring out the acceptable values of support and confidence of the rules introduced manually in B_c is out of the scope of this work.

Figure 5-3, bellow, shows an example of some rules extracted with Apriori. There are more indicators in the figure than those being cited. Indicators liked *attitude* towards school and *had_attended* pre-requisite courses. These indicators are not derived from the interactions with MOODLE. They were defined by teachers and psychologists and their values resulted from queries to the students in conjunction with analysis of their behavior (*attitude*), and import from foreign databases (*had_attended*).



```
Best rules found:

 1. submission_level=LOW approval_level=LOW 30 ==> grade=NO 30    conf:(1)
 2. posting_level=AVERAGE approval_level=AVERAGE attitude=AVERAGE 18 ==> grade=YES 18    conf:(1)
 3. approval_level=LOW attitude=LOW 17 ==> grade=NO 17    conf:(1)
 4. approval_level=HIGH attitude=AVERAGE 29 ==> grade=YES 28    conf:(0.97)
 5. submission_level=LOW attitude=LOW 28 ==> grade=NO 27    conf:(0.96)
 6. approval_level=AVERAGE had_attended=YES attitude=AVERAGE 23 ==> grade=YES 22    conf:(0.96)
 7. submission_level=AVERAGE approval_level=HIGH 21 ==> grade=YES 20    conf:(0.95)
 8. posting_level=LOW attitude=LOW 20 ==> grade=NO 19    conf:(0.95)
 9. had_attended=NO attitude=AVERAGE grade=NO 19 ==> approval_level=LOW 18    conf:(0.95)
10. approval_level=HIGH had_attended=YES attitude=AVERAGE 19 ==> grade=YES 18    conf:(0.95)
11. approval_level=AVERAGE attitude=AVERAGE 36 ==> grade=YES 34    conf:(0.94)
12. attitude=AVERAGE grade=NO 37 ==> approval_level=LOW 34    conf:(0.92)
```

Figure 5-3 Association rules derived with Apriori (support = 0.1and confidence > 0.9)

The algorithm was run interactively inside Weka GUI to produce the above set of rules. The decision makers can choose the rules they believe relevant to the diagnosing process and insert them in B_c knowledge base. Apriori can as well be run programmatically through invocation of Weka API and the rules can be immediately stored in the knowledge base. These rules can be modified or removed later by the decision makers.

As we can see from the above example, Apriori extracts relevant (and fairly expected) rules (1 and 4, for example); redundant rules (2 and 6 are extended cases of 11); and irrelevant rules for our problem (12, for instance). It is the role of teachers, psychologists and others to decide on the interest of the extracted rules. The algorithm can also find unexpected relevant rules; however, this occurs mostly when there are huge amounts of data available for analysis.

We've already pointed out that, apart from interaction variables, there are other variables, such as the formerly referenced socio-economic, cultural and behavioral ones (the *attitude* variable in **Figure**

5-3), which are interesting to help produce diagnosis. For example: it is known from the social sciences that a student from the lower classes has greater difficulties during the learning process [67]. This kind of rules cannot be inferred directly from the interactions with e-learning platforms, because usually there isn't sufficient information to do so. It should be the decision makers that, appealing to their expertise, to biographical databases, to the social sciences, etc., would create and insert them in B_c.

Finally, it must be stressed that the Data Sets may not have the necessary size to produce a reliable B_c. In this situation, the alternative might be to resort to knowledge bases B_c of similar schools or to manually insert the rules using the just mentioned criteria.

5.3 The role of SMB in the Process of Initialization of the SMs

SMB module, as we've seen on previous chapter, is the SAS component that resides and operates inside MOODLE collecting data from various sources to initialize and evolve the SMs. This module, as referenced, is sensitive to changes on MOODLE database. Whenever, during the initialization, the value of the data changes for a given student, SMB performs its job: it triggers an update to the SM, updating the attributes of predicates that compose the SM, therefore the knowledge of the system about the student.

5.4 The Role of the Technicians in the Process of Initialization of B_c

The process being described tackled the problem of extracting knowledge from interactions with MOODLE assuming that the volume of information available was enough to extract reliable rules. The technicians should decide if that volume of information is sufficient or not. The criteria to decide the minimum volume of transactions are out of scope of this work. The technicians should run periodically the Data Mining process to evaluate the stability of the beliefs in B_c. It is expected that, above a given volume of information, the beliefs set tends to stability.

As pointed out above, the technicians can, whenever they feel necessary, introduce or change beliefs directly in B_c.

In the current stage of the implementation of the system, the main method for initializing the SMs is the one referred in **5.1.1**.

The method for initializing B_c that is being used is a blend of Data Mining, like in **5.2**, and manual insertion of predicates in the knowledge base.

5.5 Conclusion

The SM can be initialized through import of data from external databases, through forms presented to the students, through default assumptions or through a combination of these methods. After that, we have a first version of the SM. As for the knowledge base B_c, it is initialized through a knowledge extraction process, also called Data Mining, combined with manual insertion of beliefs.

In the SAS architecture depicted on **Figure 4-1** the entity responsible for initializing the SMs is the SMB module. The knowledge base B_c must be initialized by the school technicians (teachers, psychologists, system administrators or others).

6 Running the Student Assessment System

In order to evaluate the operation of the SAS, we've proposed ourselves to analyse, for a given course, the results of applying the system to real world data. We've decided to compare the diagnoses produced by SAS for former students of the Shoe Modelling course with the situations that really happened (grade or not grade) on that course.

This chapter shows how to run the SAS against a SM and a B_c knowledge base and what the meaning of the output is. It also discusses the results achieved by SAS.

6.1 Evaluating Student Learning Paths

To test the evaluation of student learning paths, we've used information of former students of Shoe Modelling course collected in two moments of the learning process: half way and at the end of the course. These students never used MOODLE. But their performance was measured by the teachers based on the following parameters:

- Submission of work assignments, given by *number_of_student_submissions/number_assignments*. We have converted this to a discrete range {LOW, AVERAGE, HIGH}.

- Engagement level, meaning the participation on discussions and interventions during the classes. It was also converted to {LOW, AVERAGE, HIGH}.
- Approval on tests, the *number_of_student_approvals/number_of_quizzes*, converted to {LOW, AVERAGE, HIGH}.
- School attitude: this indicator models students' reported interest on school and on their classes; social sciences research tend to demonstrate that high achievers show more interest in learning [68] than low achievers; this interest has implications on such items like punctuality, conduct and absenteeism (absence from school). Also discrete values {LOW, AVERAGE, HIGH}.
- Grade: the real grading results: YES or NO values.

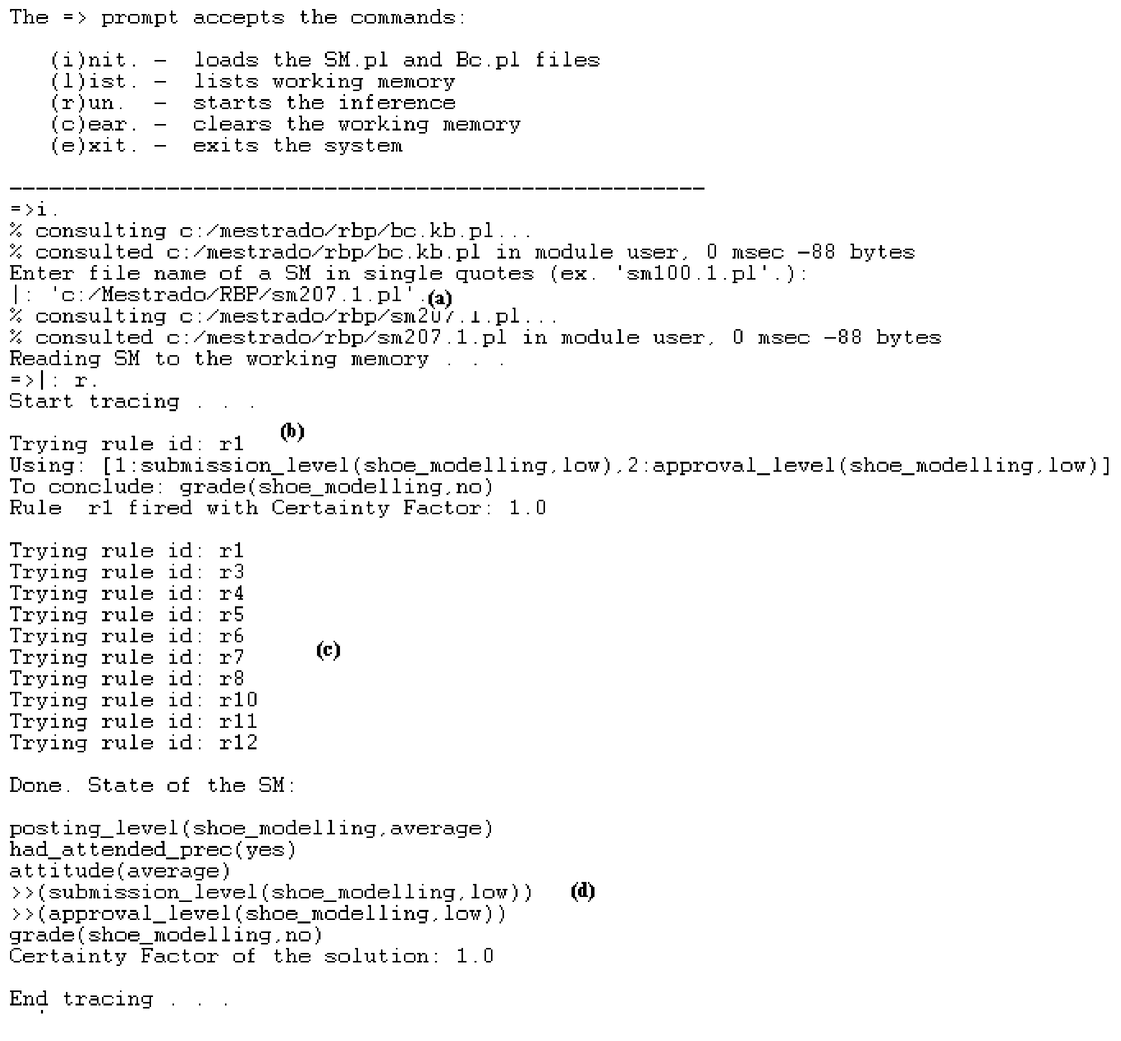
Then, we have chosen a set of attributes (already well known to us) similar to those above, but whose values came forth from interactions with MOODLE. These attributes are:

- *submission_level*: given by the *number_of_student_submissions/number_ assignments*, converted to a discrete range {LOW, AVERAGE, HIGH}.
- *posting_level*: posting on the course forum, a measure similar to engagement level, meaning the *number_of_student_posts/ number_of_posts*, also converted to {LOW, AVERAGE, HIGH}.
- *approval_level*: approval on tests, *number_of_student_approvals/number_of_quizzes*, converted to {LOW, AVERAGE, HIGH}.
- *attitude*: attitude towards learning and its correlates like punctuality, conduct and absenteeism. Also a discrete set {LOW, AVERAGE, HIGH}.

The *grade* attribute, i.e., the diagnosis we want is also a discrete set with YES or NO values.

Next, we've run the Apriori algorithm found in Weka against MOODLE database to find associative rules between *grade* and the other attributes. Note that the attributes above had to be added to MOODLE database, as we've explained in **4.2.1**. The result was a knowledge base, B_c, composed of rules extracted with Apriori, as well as rules added manually, as shown in **Figure 4-5.**

With these SMs we've run the SMD of the SAS against the knowledge base B_c of **Figure 4-5**, and compared the predictions of the system with the results really achieved by the students. Next figure shows how to run SMD to output a diagnosis. Previously we had to load SMD into the Prolog interpreter and call it. We've defined a predicate `shell/0` that starts SMD and displays the menu on the figure.



```
The => prompt accepts the commands:

   (i)nit. -  loads the SM.pl and Bc.pl files
   (l)ist. -  lists working memory
   (r)un.  -  starts the inference
   (c)ear. -  clears the working memory
   (e)xit. -  exits the system

----------------------------------------------------
=>i.
% consulting c:/mestrado/rbp/bc.kb.pl...
% consulted c:/mestrado/rbp/bc.kb.pl in module user, 0 msec -88 bytes
Enter file name of a SM in single quotes (ex. 'sm100.1.pl'.):
|: 'c:/Mestrado/RBP/sm207.1.pl'.
% consulting c:/mestrado/rbp/sm207.1.pl...
% consulted c:/mestrado/rbp/sm207.1.pl in module user, 0 msec -88 bytes
Reading SM to the working memory . . .
=>|: r.
Start tracing . . .

Trying rule id: r1
Using: [1:submission_level(shoe_modelling,low),2:approval_level(shoe_modelling,low)]
To conclude: grade(shoe_modelling,no)
Rule  r1 fired with Certainty Factor: 1.0

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Trying rule id: r12

Done. State of the SM:

posting_level(shoe_modelling,average)
had_attended_prec(yes)
attitude(average)
>>(submission_level(shoe_modelling,low))
>>(approval_level(shoe_modelling,low))
grade(shoe_modelling,no)
Certainty Factor of the solution: 1.0

End tracing . . .
```

Figure 6-1 Running the SMD of the SAS

From that on it's just following the instructions.

As we've said before, in **4.3.3**, and it is expressed on the figure above, **(a)** we've started by loading B_c knowledge base and SM into SMD working memory. In this example the SM is that of the student whose ID is 207 (**Figure 6-2** below).

```
SM(207) = [submission_level(shoe_modelling, low),
posting_level(shoe_modelling,average),
approval_level(shoe_modelling,low),
had_attended_prec(yes),
attitude(average)]
```

Figure 6-2 SM of student 207

Next, **(b)**, we run SMD, which tried all the rules in B_c, **(c)**, to diagnose, with a Certainty Factor of 1, **(d)**, that the student 207 will not grade on Shoe Modelling course.

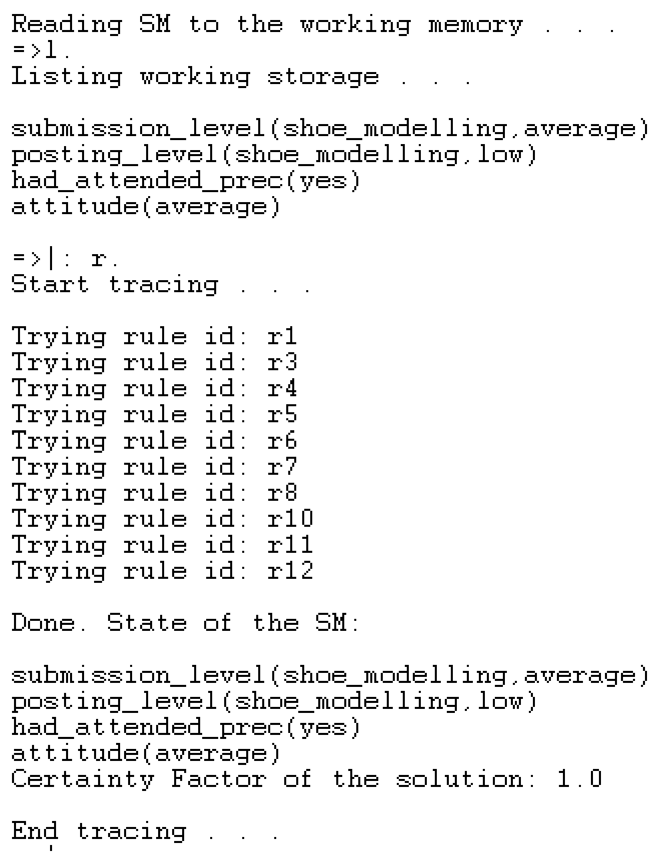
Before analysing the results, and for the sake of clarity, let's say a couple of words about the execution of the program. We've classified SMD's runs in three types, based on the results returned:

1. The type where there is no Action that matches any Condition, therefore no resulting diagnosis;
2. The type where there are matching Actions and a resulting diagnosis;
3. The type where there are matching Actions but no diagnosis can be outputted.

Now, let's briefly discuss each of the types.

6.1.1 There is no Action that matches any Condition

The type of run we are talking about here is depicted on the next figure.



```
Reading SM to the working memory . . .
=>l.
Listing working storage . . .

submission_level(shoe_modelling,average)
posting_level(shoe_modelling,low)
had_attended_prec(yes)
attitude(average)

=>|: r.
Start tracing . . .

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Trying rule id: r12

Done. State of the SM:

submission_level(shoe_modelling,average)
posting_level(shoe_modelling,low)
had_attended_prec(yes)
attitude(average)
Certainty Factor of the solution: 1.0

End tracing . . .
```

Figure 6-3 No Action matches any Condition

There is no rule in B_c whose Condition side matches the facts in the working storage (SM). In this case the SAS can't give a diagnosis. One has to insert new rules in order to establish the condition(s) that could solve the SM (i.e., produce a diagnosis).

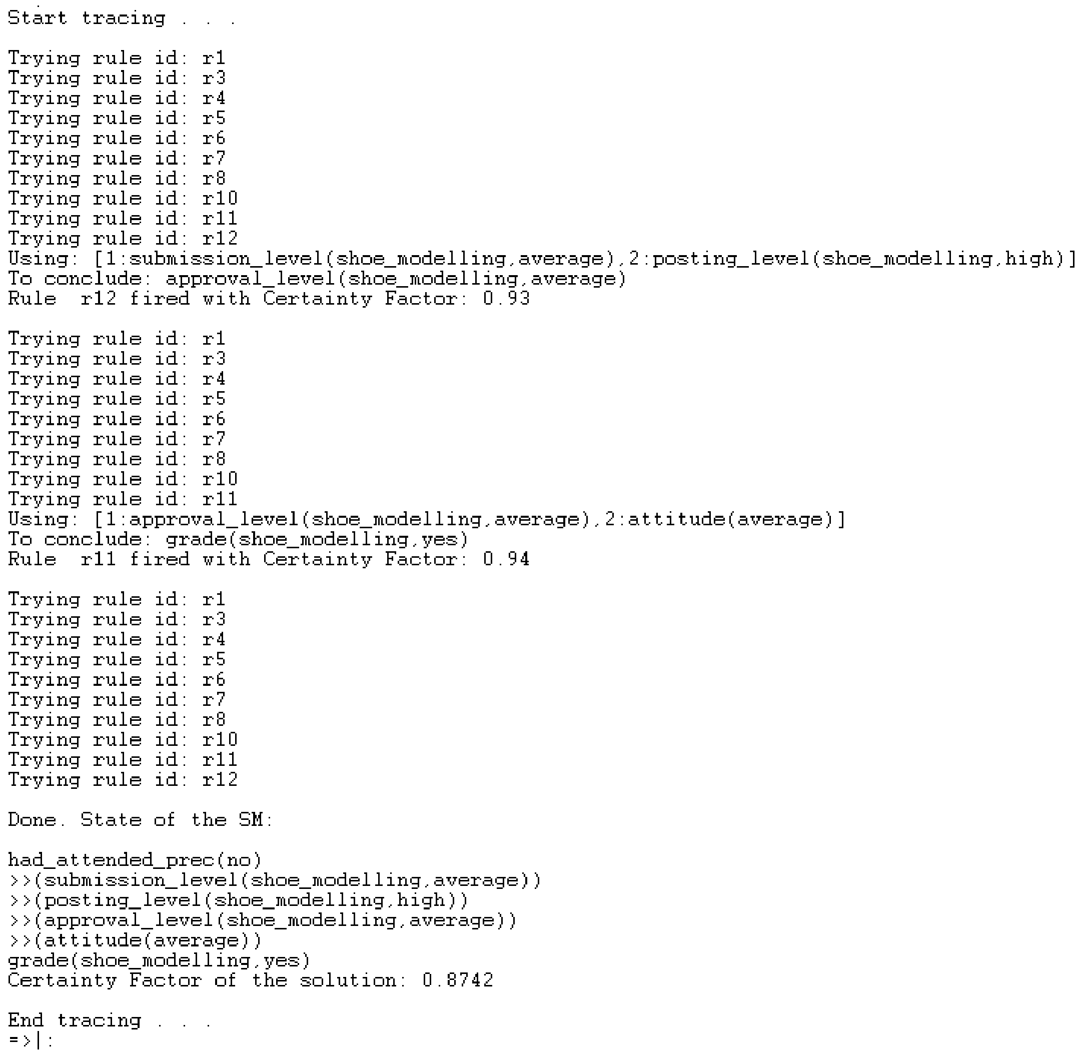
6.1.2 There are Matching Actions and a Diagnosis is found

This type of run is like the one depicted on the **Figure 6-1** above and in **Figure 6-4** below. The Condition side of one, or more, of the rules matches one, or a combination of, the facts in the working storage (SM). In **Figure 6-1** the diagnosis is immediately found after the first run and the Certainty Factor is the one associated with that rule. In that case, the rule that matched was

$$r1{:}[1{:}submission\_level(X,low),\ 2{:}approval\_level(X,low)] \Rightarrow [retract(all),\ assert(grade(X,no))]{:}($$

The Certainty Factor of the diagnosis is that of the rule (1.0, in the example).

The example in **Figure 6-4** shows a fact that is derived from the knowledge base and then is used as part of a match to find a diagnosis. This is the most common case because usually there are relations in B_c between attributes of the SMs other then the diagnostic attribute (*grade*, in our case).



```
Start tracing . . .

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Trying rule id: r12
Using: [1:submission_level(shoe_modelling,average),2:posting_level(shoe_modelling,high)]
To conclude: approval_level(shoe_modelling,average)
Rule  r12 fired with Certainty Factor: 0.93

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Using: [1:approval_level(shoe_modelling,average),2:attitude(average)]
To conclude: grade(shoe_modelling,yes)
Rule  r11 fired with Certainty Factor: 0.94

Trying rule id: r1
Trying rule id: r3
Trying rule id: r4
Trying rule id: r5
Trying rule id: r6
Trying rule id: r7
Trying rule id: r8
Trying rule id: r10
Trying rule id: r11
Trying rule id: r12

Done. State of the SM:

had_attended_prec(no)
>>(submission_level(shoe_modelling,average))
>>(posting_level(shoe_modelling,high))
>>(approval_level(shoe_modelling,average))
>>(attitude(average))
grade(shoe_modelling,yes)
Certainty Factor of the solution: 0.8742

End tracing . . .
=>|:
```

Figure 6-4 Action is found after two runs

In this example, the rule *r12* is fired to conclude, with CF = 0.94, that the approval level of the student in shoe modelling is average. Then the rule *r11* is fired to diagnose grading with CF = 0.8742. The CF of the solution is the product of all CFs.

6.1.3 There are matching Actions but no diagnosis is found

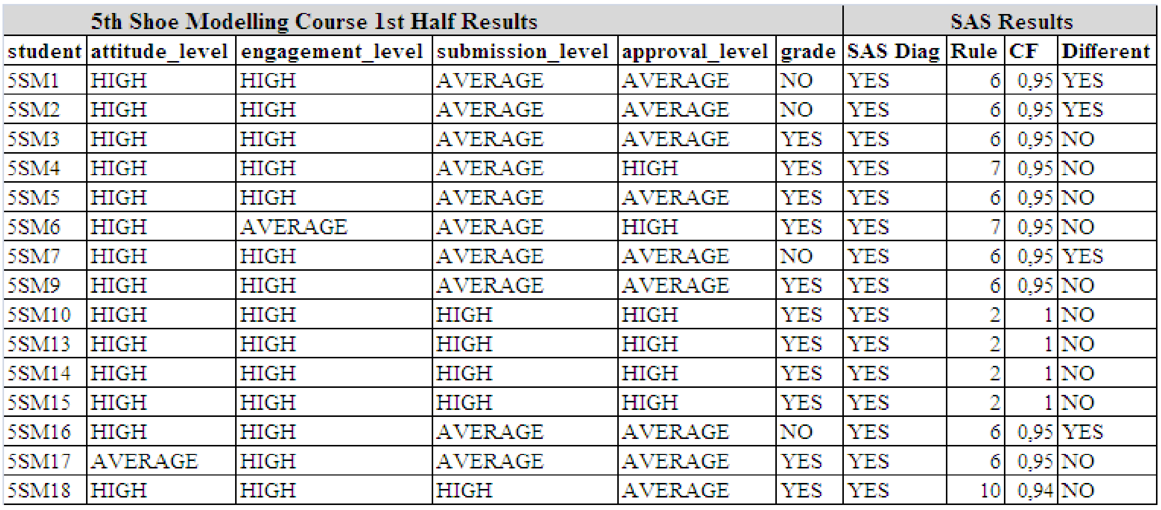
This case is similar to that in **Figure 6-4** except for the fact that there are no matching Conditions that could produce a diagnosis at the end. This means that SMD can find several new facts, resulting from relations between the attributes, but none of the facts, the ones already in B_c and the new ones found, match any Condition that had a diagnosis (*grade*) as part of its Action.

6.2 Discussion of the results

As said before, to test our model we've used results from former students of the Shoe Modelling course. The purpose is to build a case study to verify how well the system applies to the real world.

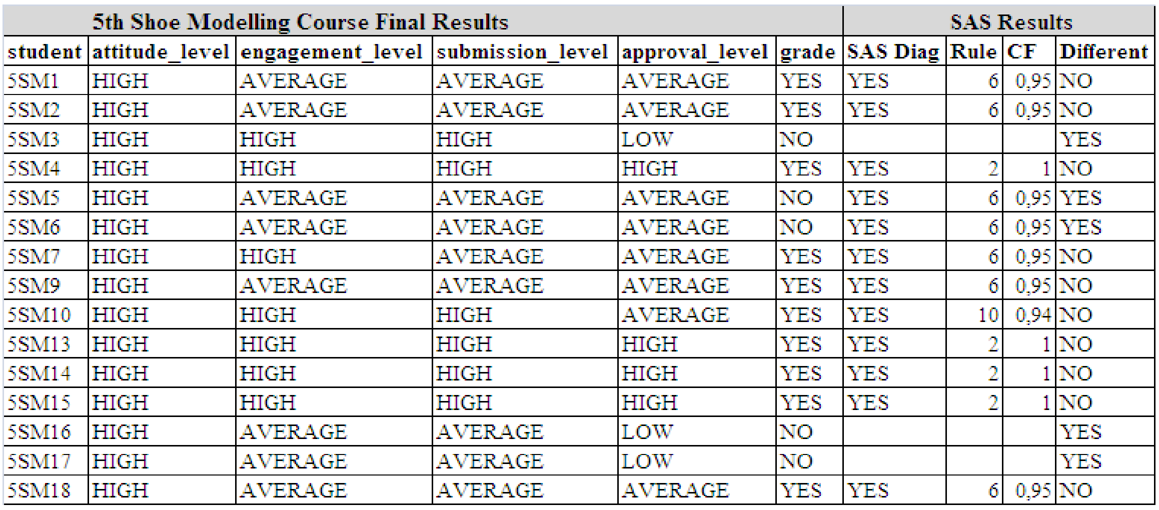
6.2.1 Result tables

Next figures present tables with the results obtained by the students of two different classes (the 5th and the 10th) of the Shoe Modelling Course and the ones obtained by running SAS SMD inference engine.



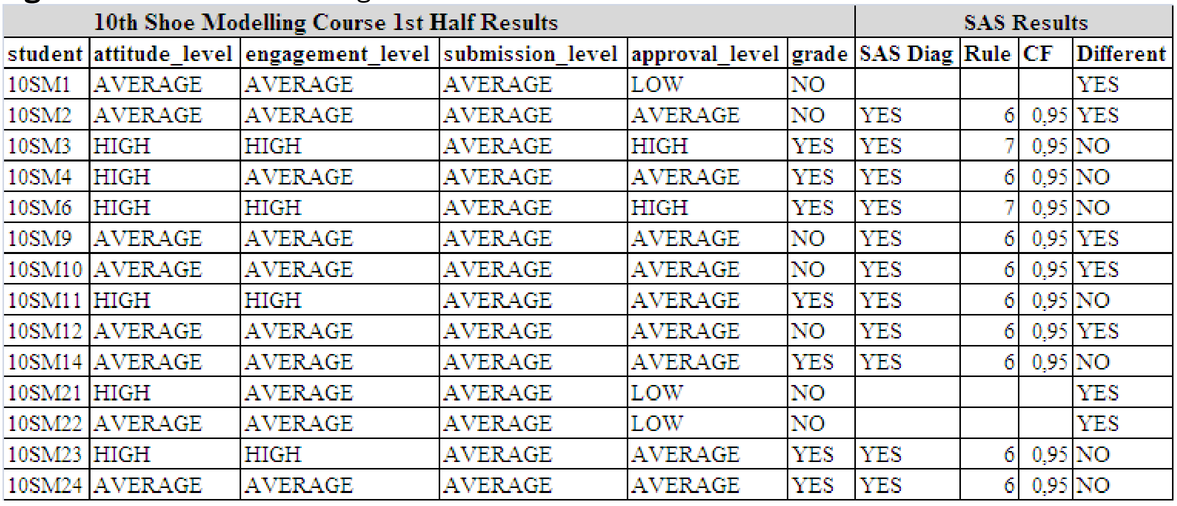
5th Shoe Modelling Course 1st Half Results						SAS Results			
student	attitude_level	engagement_level	submission_level	approval_level	grade	SAS Diag	Rule	CF	Different
5SM1	HIGH	HIGH	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
5SM2	HIGH	HIGH	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
5SM3	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM4	HIGH	HIGH	AVERAGE	HIGH	YES	YES	7	0,95	NO
5SM5	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM6	HIGH	AVERAGE	AVERAGE	HIGH	YES	YES	7	0,95	NO
5SM7	HIGH	HIGH	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
5SM9	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM10	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM13	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM14	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM15	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM16	HIGH	HIGH	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
5SM17	AVERAGE	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM18	HIGH	HIGH	HIGH	AVERAGE	YES	YES	10	0,94	NO

Figure 6-5 5th Shoe Modelling Course 1st half results and SAS results



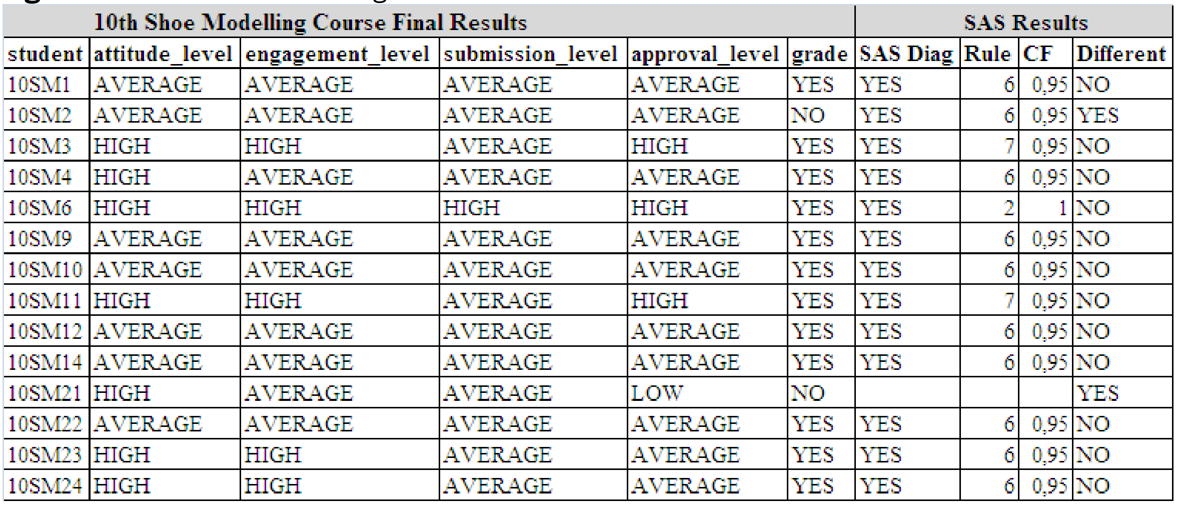
5th Shoe Modelling Course Final Results						SAS Results			
student	attitude_level	engagement_level	submission_level	approval_level	grade	SAS Diag	Rule	CF	Different
5SM1	HIGH	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM2	HIGH	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM3	HIGH	HIGH	HIGH	LOW	NO				YES
5SM4	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM5	HIGH	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
5SM6	HIGH	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
5SM7	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM9	HIGH	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
5SM10	HIGH	HIGH	HIGH	AVERAGE	YES	YES	10	0,94	NO
5SM13	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM14	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM15	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
5SM16	HIGH	AVERAGE	AVERAGE	LOW	NO				YES
5SM17	HIGH	AVERAGE	AVERAGE	LOW	NO				YES
5SM18	HIGH	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO

Figure 6-6 5th Shoe Modelling Course final results and SAS results



10th Shoe Modelling Course 1st Half Results						SAS Results			
student	attitude_level	engagement_level	submission_level	approval_level	grade	SAS Diag	Rule	CF	Different
10SM1	AVERAGE	AVERAGE	AVERAGE	LOW	NO				YES
10SM2	AVERAGE	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
10SM3	HIGH	HIGH	AVERAGE	HIGH	YES	YES	7	0,95	NO
10SM4	HIGH	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM6	HIGH	HIGH	AVERAGE	HIGH	YES	YES	7	0,95	NO
10SM9	AVERAGE	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
10SM10	AVERAGE	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
10SM11	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM12	AVERAGE	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
10SM14	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM21	HIGH	AVERAGE	AVERAGE	LOW	NO				YES
10SM22	AVERAGE	AVERAGE	AVERAGE	LOW	NO				YES
10SM23	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM24	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO

Figure 6-7 10th Shoe Modelling Course 1st half results and SAS results



10th Shoe Modelling Course Final Results						SAS Results			
student	attitude_level	engagement_level	submission_level	approval_level	grade	SAS Diag	Rule	CF	Different
10SM1	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM2	AVERAGE	AVERAGE	AVERAGE	AVERAGE	NO	YES	6	0,95	YES
10SM3	HIGH	HIGH	AVERAGE	HIGH	YES	YES	7	0,95	NO
10SM4	HIGH	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM6	HIGH	HIGH	HIGH	HIGH	YES	YES	2	1	NO
10SM9	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM10	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM11	HIGH	HIGH	AVERAGE	HIGH	YES	YES	7	0,95	NO
10SM12	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM14	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM21	HIGH	AVERAGE	AVERAGE	LOW	NO				YES
10SM22	AVERAGE	AVERAGE	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM23	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO
10SM24	HIGH	HIGH	AVERAGE	AVERAGE	YES	YES	6	0,95	NO

Figure 6-8 10th Shoe Modelling Course final results and SAS results

As we can see, the tables present a relation between the classifications given by the teachers in two moments of evaluation, at half time and at the end of the course. The meaning of *grade* at half time is just a measure of the performance of the student till then.

6.2.2 Difference and Error in the Diagnosis

The last column of the tables (Different) denotes the existence of differences between the results of the students and the diagnosis of the SAS. When there is a diagnosis in *SAS_Diag* column (**6.1.2** type of run) and the diagnosis of SAS is unlike the results obtained (in *grade* column), one considers that we are in the presence of **a difference that is an error** of the SAS. On the other hand, an empty cell in *SAS_Diag* column indicates that the SAS-SMD could not infer a diagnosis from the rules in B_c (**6.1.1** and **6.1.3** types of run). We consider this not as an error but **just as a difference**. So, a difference occurs whenever the results of SAS are not equal to the results of the student on the course:

$$differece(s) : grade(s) \neq SAS\_Diag(s) \qquad \textbf{(5)}$$

And an error for a given studen is a difference such as:

$$error(s) : difference(s) \wedge \exists SAS\_Diag(s) \qquad \textbf{(6)}$$

Next figures show some graphics of the evolution of differences and errors on the diagnosis of SAS-SMD inference engine for the 5th and 10th classes of Shoe Modelling Course.

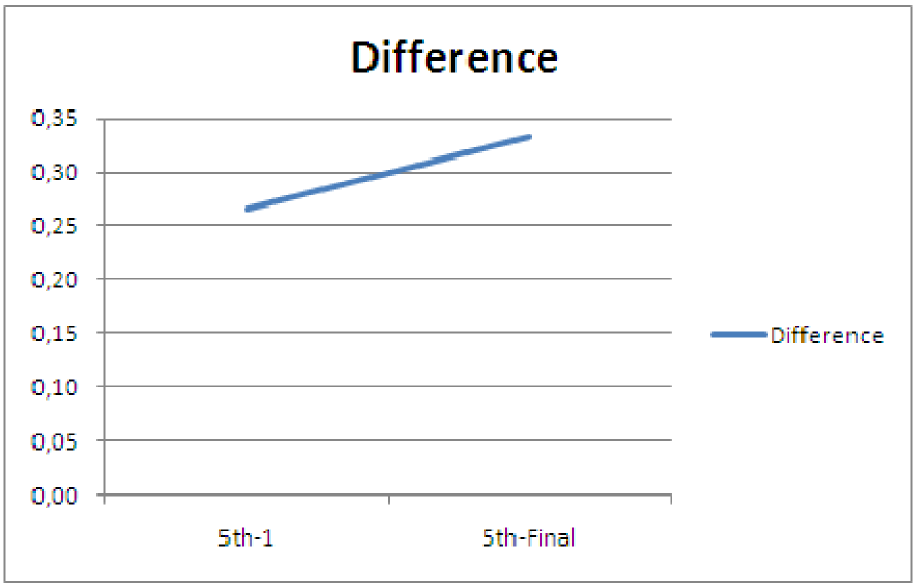


Figure 6-9 Evolution of the difference of SAS diagnosis for 5th Shoe Modelling Course

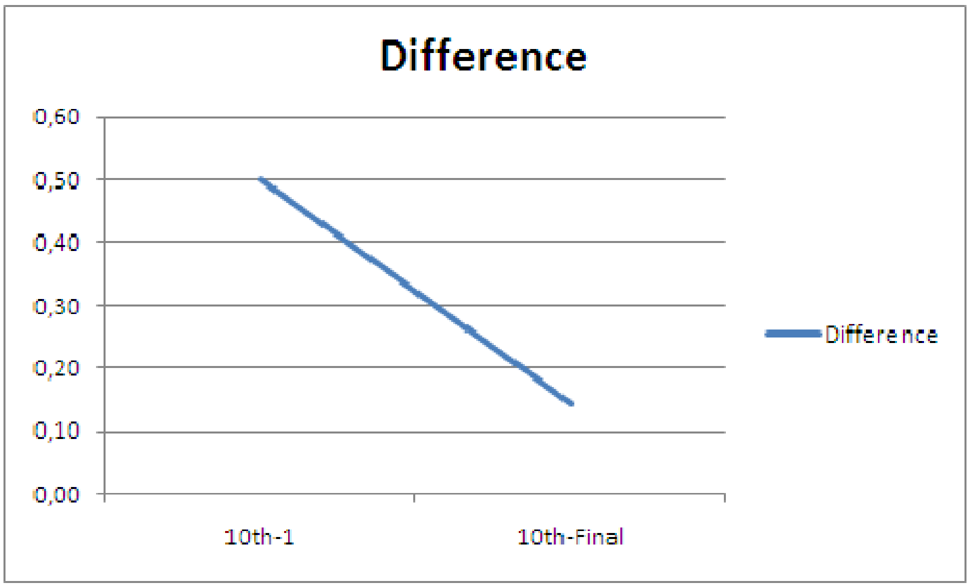


Figure 6-10 Evolution of the difference of SAS diagnosis for 10th Shoe Modelling Course

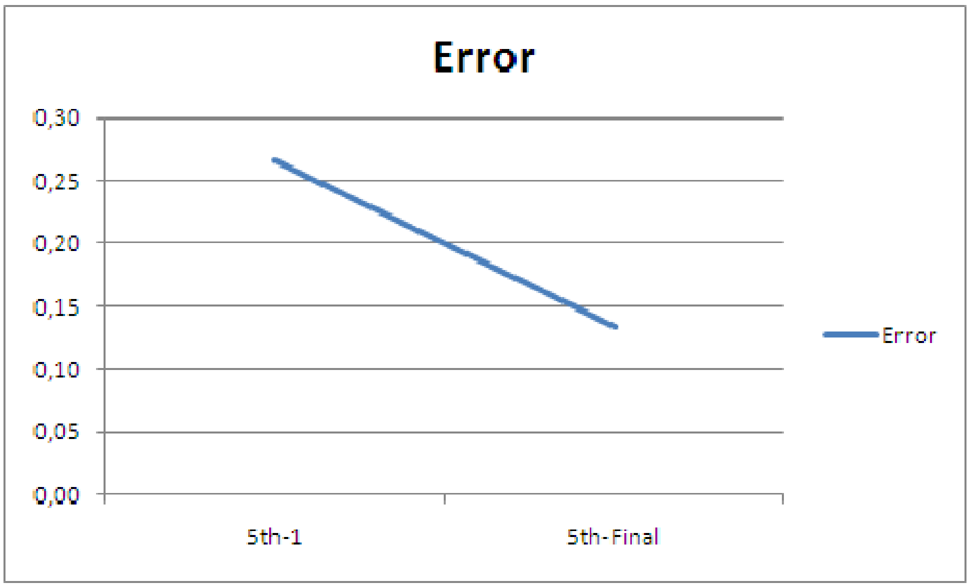


Figure 6-11 Evolution of the error of SAS diagnosis for 5th Shoe Modelling Course

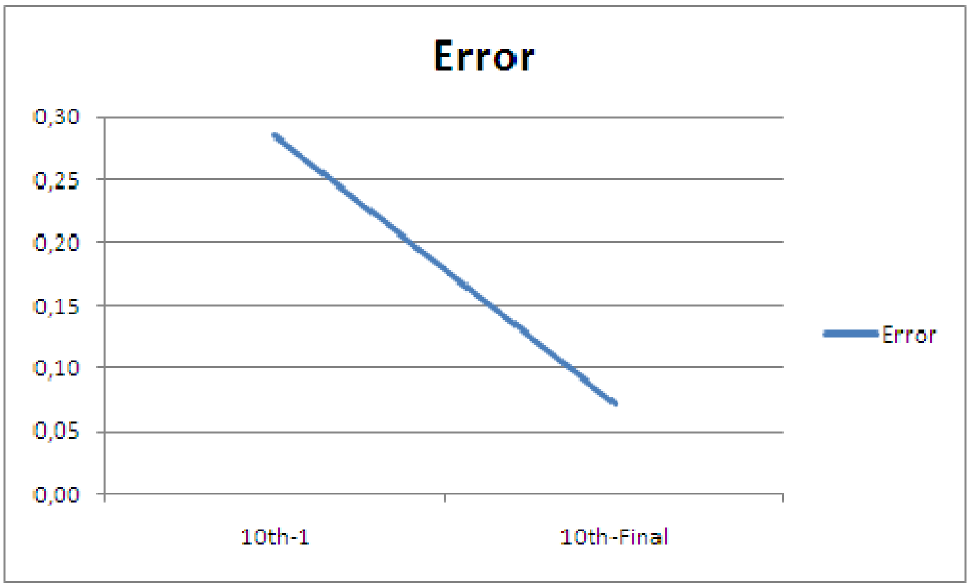


Figure 6-12 Evolution of the error of SAS diagnosis for 10th Shoe Modelling Course

As we can see, the differences and errors in SAS diagnosis are small. The exception is the 1st half difference of the 10th class which is a combination of impossibility to produce a diagnosis for three students with wrong diagnosis for four students. Other oddity is the increasing of difference in the 5th class, although the error had decreased; this is explained by the appearing of facts not represented in conditions in B_c. But, in general the trend of the evolution of the differences and errors on SAS diagnosis is to decrease from the first half to the final moments of evaluation of the students. As the time goes, the teacher acquires more information about the student (its own knowledge base B_c, so to speak). At the same time, the classifications given by the SAS tend to be similar to those of the teachers. This is an indication that the knowledge of the system (SM and B_c) is really useful to produce diagnosis, as it converges to that of the teacher.

Another interesting fact is that the teacher does not always pass a student although she has the same values of the indicators as another that have been passed by the same teacher. SAS does not behave that way because the rules triggered, in the current implementation, are always the same. For example, in the 1st half results of the 5th Course (**Figure 6-5**), the students 5SM2 and 5SM3 both have the same values of the indicators; but the former had graded and the latter did not. SAS "passes" both students. On one hand this indicates that there are hidden variables on the teacher's decision not considered in our model; on the other this also explains the need for a certainty factor. Confronted with these cases, SAS-SMD applies rule ***r6*** and diagnosis the student as grading with ***CF = 0.95***.

Finally, it has been said that the SMD does not always returns a diagnosis. When there are no rules whose Condition side applies to facts of the SM (**6.1.1**) or when none of the Actions triggered are functions of *grade* (**6.1.3**), the system is not able to output a diagnosis. This is the case, for instance, on 5th Course final results, students 5SM3, 5SM16 and 5SM17 (**Figure 6-6**). What to do in such circumstances? Our bet is to submit the missing rules to strong analysis and, eventually, evolve the knowledge base with those rules that the decision makers feel that make sense (**5.2** above).

6.3 Conclusion

To put the system into test, we've run SAS with data from former students of Shoe Modelling course. This way, we've built a case study that makes it possible to derive some conclusions about the behaviour of the system in real world situations.

The analysis of the results of the case study points out clearly that SAS is very effective in diagnosing the student learning paths when all the relevant predicates are present in SM and all the necessary rules are present in B_c. Its error is usually low and tends to decrease as the courses develop.

7 Conclusions and Future Work

The particular nature of the learning process demands grounded decisions taken on time, to be able to detect problems when they arise and avoid failures. There is a big price to pay for student failure. Therefore, schools should acquire tools to help forecast student learning problems. This work is an answer to that need, as it provides the means to assess the students' learning process and to detect problems before they become critical.

7.1 Synthesis of the SAS Project

A SAS, as the one described here, is able to build and evolve SMs from the logs of an e-learning platform and from a knowledge base of beliefs (or production rules). It is also able to output diagnosis of student models with a given QoI, and with a CF that is the composition of the CFs of the beliefs. This is the confirmation of the hypothesis we've put at the beginning of this work.

Also, the goals we've proposed to attain were all accomplished, namely:

- The system is able to build Student Models from the interactions of the students with MOODLE. These models have proved useful to produce forecasts of students' learning processes.
- The knowledge representation as ELP productions is capable of handling uncertainty and representing the beliefs, or rules, in a way that is understood and manageable by the system;

- The theoretical framework proposed has proved a firm ground to support the SAS;
- The architecture of the system has shown effective to model the various parts of the system;
- The choices made during the development (tools and technologies) were rewarded with a functioning and useful system that achieves the objectives proposed.

This system has been put to test on a professional school with the results we've just shown. In brief, we saw a convergence on the results of the diagnosis of the system and those of the teachers. There are indeed differences between the diagnosis of the system and the diagnosis of the teachers. These differences are mainly due to lack of knowledge of the system and to the subjectivity of the decision process of the teachers.

7.2 Relevant Work

As we've said in the beginning, this work was funded by CFPIC and was developed in the environment of this Professional School. Its development has been subject of several publications, witnessing its evolution, namely

- Almeida P., Novais P., Costa E., Rodrigues M., Neves J.: **Artificial Intelligence Tools for Student Learning Assessment in Professional Schools**, in Proceedings of the 7th European Conference on e-Learning, Cyprus, November, ISBN 978-1-906638-22-1, pp 17-28, 2008 (Indexed by ISI Web of Science).
- Almeida P., Novais P., Neves J.: **Prediction Tools for Student Learning Assessment in Professional Schools**, in Proceedings of the ESM 2008 - The 22nd annual European Simulation and Modelling Conference, Le Havre, France, October, ISBN 978-90-77381-44-1, pp 121-129, 2008 (Indexed by ISI Web of Science).
- Almeida, P.: **Metodologias para Inicialização dos Modelos de Aluno e da Base de Conhecimento de um Sistema de Seguimento de Alunos**, Technical Report, University of Minho, Braga, Portugal, January 2009.
- Almeida, P., Novais, P.: **A Student Assessment System - Decision Support for Student Models Diagnosing**, Article submitted to the Wiley InterScience Journal on Computer Applications in Engineering Education, September 2009.

7.3 Future Work

This work has proved to be an answer to the need the Professional Schools have to follow their students' learning careers in order to minimize the failure situations. It provides the means to assess the students' learning processes and to detect problems before they become critical. There is much space for evolution, mainly on behalf of improvements on the prediction of learning paths and on SMD inference process, as well as on easing the introduction of rules and, in general, on the system usability.

7.3.1 Improvements on Predictions of Learning Paths

By storing all the versions of the SM of a student, including the various diagnostics, we are saving the student's learning path (SLP). This allows us to go further on analysing the student's career. At any time, during a student's learning process, we can compare the similarities of the SLP of the student with the SLPs of other students that had attended the same course before and try to forecast her probabilities of success. We can use a technology like Case Based Reasoning (CBR) [69] to produce such forecasts [70].

For example, let us consider the following diagnostic attributes for a student on a course C (with a certain correlation coefficient R):

SAS_Diag at half time, DH: real

SAS_Diag at the end of the course, DF: real

and the corresponding regression equation[6], or prediction model:

$$Y = a + b_1 \times DH + b_2 \times DF$$

where *a* is the intercept coefficient, b_i the slopes or regression coefficients and *Y* is the grade prediction probability for the student on the course. Those coefficients may be a subset of those used by Kruck and Lending [71]. Furthermore, one could use the prediction model itself to compute the similarities on the retrieve step of the CBR reasoning process.

[6] In the social sciences, multiple regression procedures are very widely used in research.

7.3.2 Enhancements to SMD

In its current state, SMD is a simple forward chaining system. More advanced forward chaining systems differ in two main aspects:

- More sophisticated rule selection when many rules match the current working storage;
- Better performance.

The current rule selection strategy of SMD is simply to pick the first rule that matches. If several rules match, there might be other optimal choosing strategies. For example, we could pick the rule that matched the most recently asserted facts. Or we could pick the rules based on CF. The definition of CF, as we've said before, is the confidence of a rule found by Apriori algorithm, for a given support of B_c. The last strategy is the one we favour most and is the one we're using: We're just ordering the rules by CF. Whatever the chosen strategy, it would change the inference pattern of the system and the output of the inference might make more sense. It is clear that to evolve the system one needs some sort of "conflict resolution" methodology.

SMD uses a brute force approach to pattern matching, since at each cycle of the system it tries all of the rules against working memory. There are various indexing schemes which can be used to allow for faster picking of rules that match working memory. Some high-end expert systems use the Rete match algorithm [72] to optimize performance. This is an indexing scheme which allows for rapid matching of left hand side rule patterns against working storage elements. Previous match information is saved on each cycle, so the system avoids redundant calculations.

7.3.3 User Interface for Rules' Introduction

The main drawback of using Prolog to create rules in B_c is that users are less familiarized with logical programming than with procedural programming. There is also some lack of propositional and predicate logic knowledge among users, who have some difficulty understanding the rules, as well as developing new ones. A graphical interface to introduce the rules in the knowledge base would certainly be welcomed by the users.

Bibliography

[1] Martins A., Pardal L., Dias C.: **Ensino Técnico e Profissional: Natureza da Oferta e da Procura,** Revista Interacções N°. 1, Escola Superior de Educação de Santarém, pp. 77-97, 2005.

[2] Plano Tecnológico da Educação, **http://www.escola.gov.pt/inicio.asp**, accessed on October 2009.

[3] Programa Novas Oportunidades, **http://www.novasoportunidades.gov.pt/**, accessed on October 2009

[4] Consortium for School Networking: Taking **TCO to the Classroom, A School Administrator's Guide To Planning for the Total Cost of New Technology**, July 2001.

[5] McKinsey & Company, Inc.: **Connecting K-12 Schools to the Information Superhighway**, 1995. Available at **http://www.uark.edu/mckinsey**.

[6] Almeida P., Novais P., Costa E., Rodrigues M., Neves J.: **Artificial Intelligence Tools for Student Learning Assessment in Professional Schools**, in Proceedings of the 7th European Conference on e-Learning, Cyprus, November, ISBN 978-1-906638-22-1, pp 17-28, 2008.

[7] Bates, A. and Bates, T.: **Technology, e-learning and distance education**, Second Edition, Routledge, 2005.

[8] Wikipedia, **http://en.wikipedia.org/wiki/Electronic_learning**, accessed on October 2009.

[9] Nagy, A.: **The Impact of E-Learning**, in: Bruck, P.A.; Buchholz, A.; Karssen, Z.; Zerfass, A. (Eds), in E-Content: Technologies and Perspectives for the European Market, Berlin, Springer-Verlag, pp.79-96, 2005.

[10] Nichols, M.: **E-Learning in context**, in http://akoaotearoa.ac.nz/sites/default/files/ng/group-661/n877-1—e-learning-in-context.pdf, 2008.

[11] Andes Physics Tutor, **http://www.andestutor.org/**, accessed on October 2009.

[12] MOODLE: **http://moodle.org/**, accessed on October 2009.

[13] Rodrigues, M.: **Proposta de uma Framework para Desenvolvimento de Sistemas Tutores Inteligentes**, MSc Dissertation, Information Systems Department, University of Minho, Braga, Portugal, June 2007.

[14] Romero, C., Ventura S., Espejo, P. and Hervás, C.: **Data Mining Algorithms to Classify Students,** Computer Science Department, Córdoba University, Spain, 2005.

[15] Romero, C., Ventura, S., Garcia, E.: **Data mining in course management systems: MOODLE case study and tutorial**, Computers & Education, Elsevier Science, 2007.

[16] Garcia, E., Romero, C., Ventura, S., Calders, T.: **Drawbacks and solutions of applying association rule mining in learning management systems**, Proceedings of the International Workshop on Applying Data Mining in e-Learning, Crete, Greece, September 17-20, 2007.

[17] Calvo-Flores, M., Galindo, E., Jiménez, M., and Piñeiro, O.: **Predicting students' marks from Moodle logs using neural network models**, in Current Developments in Technology-Assisted Education, pp. 586-590, 2006.

[18] O'Brien, R.: **An overview of the methodological approach of Action Research,** in Roberto Richardson (Ed.), Theory and Practice of Action Research, João Pessoa, Brazil: Universidade Federal da Paraíba, 2001. English version: http://www.web.ca/~robrien/papers/arfinal.html, accessed on October 2009.

[19] XML, **http://www.w3.org/XML/**, accessed on October 2009.

[20] SOAP, **http://www.w3.org/TR/soap/**, accessed on October 2009.

[21] AICC, **http://www.aicc.org/**, accessed on October 2009.

[22] SCORM, **http://www.adlnet.gov/Technologies/scorm/default.aspx**, accessed on October 2009.

[23] IMS, **http://www.imsglobal.org/**, accessed on October 2009.

[24] IEEE, **http://www.ieee.org/portal/site**, accessed on October 2009.

[25] Cognitive Design Solutions, **http://www.cognitivedesignsolutions.com/**, accessed on October 2009.

[26] SQL, **http://www.sql.org/**, accessed on October 2009.

[27] TCP/IP, RFCs available at **http://www.ietf.org/**, accessed on October 2009.

[28] HTTP, **http://www.w3.org/Protocols/**, accessed on October 2009.

[29] Van Lehn K. (2006): **The behavior of tutoring systems**, International Journal of Artificial Intelligence in Education, 16 (3), pp. 227-265.

[30] Liu, X., Saddik, A. and Georganas N.: **An Implementable Architecture of an E-Learning System**, Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering, CCECE, Vol. 2, pp. 717-720, Montreal, May 2003.

[31] Papert, S. and Harel, I.: **Constructionism**, First Chapter, Ablex Publishing Corporation, 1991.

[32] WebCT/Blackboard, **http://www.blackboard.com/**, accessed on September 2009.

[33] Unix, **http://www.unix.org/**, accessed on October 2009.

[34] Linux, **http://www.linux.org/,** accessed on October 2009.

[35] Windows, **http://www.microsoft.com/en/us/default.aspx**, accessed on October 2009.

[36] Mac OS X, **http://www.apple.com/macosx/**, accessed on October 2009.

[37] PHP 5, **http://www.php.net/**, accessed on October 2009.

[38] MySQL: **http://www.mysql.com/**, Version 5.0, 2008.

[39] PostgreSQL, **http://www.postgresql.org/**, accessed on October 2009.

[40] Oracle, **http://www.oracle.com/index.html**, accessed on October 2009.

[41] SQLServer DBMS, **http://www.microsoft.com/sqlserver/2008/en/us/default.aspx**, accessed on October 2009.

[42] ODBC, **http://msdn.microsoft.com/en-us/library/ms710252(VS.85).aspx**, accessed on October 2009.

[43] Foster, R., Fletcher, J.: **Modeling the User for Education, Training, and Performance Aiding**, IDA Document D-2920, Institute for Defense Analysis, Alexandria, Virginia, 2003.

[44] Bayes, T: **An Essay towards solving a Problem in the Doctrine of Chances,** in Philosophical Transactions of the Royal Society of London Nr. 53, pp. 370–418, 1763.

[45] Clancey, W.: **Qualitative Student Models**, in Annual Review of Computer Science, Vol.1, pp. 381-450, May 1986.

[46] Arroyo, I., Murray, T., Woolf, B. and Beal C.: **Inferring unobservable learning variables from students' help seeking behaviour**, Workshop Proceedings of International Conference on Intelligent Tutoring Systems, pp. 782-784, 2004.

[47] Jonsson, A., Johns, J., Mehranian, H., Arroyo, I., Woolf, B., Barto, A., Fisher, D. and Mahadevan, S.: **Evaluating the feasibility of learning student models from data**, Proceedings of the Workshop on Educational Data Mining at Association for the Advancement of Artificial Intelligence (AAAI), 2005.

[48] Mislevy, R.J., Almond, R.G., Yan, D., and Steinberg, L.S.: **Bayes nets in educational assessment: Where do the numbers come from?**, Laskey and Prade, editors, Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 437-446, 1999.

[49] Ben-Gal, I.: **Bayesian Networks**, in Ruggeri, Fabrizio; Kennett, Ron S.; Faltin, Frederick W. Encyclopedia of Statistics in Quality and Reliability, John Wiley & Sons, 2007.

[50] Neves, J.: **A Logic Interpreter to Handle Time and Negation in Logic Data Bases**, in Proceedings of the ACM'84, The Fifth Generation Challenge, 1984.

[51] Way, E.C.: **Knowledge Representation and Metaphor**, Kluwer Academic Publishers, Dordrecht, Holland, 1991.

[52] Analide C., Novais P., Machado J., Neves J.: **Quality of Knowledge in Virtual Entities**, Encyclopaedia of Communities of Practice in Information and Knowledge Management, Coakes and Clarke, editors, Idea Group Reference, pp. 436-442, 2006.

[53] Ginsberg, M.L.: **Readings in Non-monotonic Reasoning**, Morgan Kauffman Publishers, Los Altos, California, EUA, 1991.

[54] Hustadt, U.: **Do we need the closed-world assumption in knowledge representation?** in *Working Notes of the KI'94 Workshop*, Saarbrüken, Germany, 1994.

[55] Parsons, S.: **Current approaches to handling imperfect information in data and knowledge bases**, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 3, pp. 353-372, 1996.

[56] Self, J.: **Formal Approaches to Student Modeling**, AAI/AI-ED Technical Report No. 92, 1994.

[57] Gelfond, M. and Lifschitz V.: **Logic Programs with Classical Negation**, in Proceedings of the International Conference on Logic Programming, 1990.

[58] Windows Server, **http://www.microsoft.com/windowsserver2008/en/us/default.aspx**, accessed on October 2009.

[59] Bratko I.: **Prolog Programming for Artificial Intelligence**, Third Edition, Addison-Wesley, 2001, pp. 355-358 and pp. 631-644.

[60] Brna, P.: **Prolog Programming – A First Course**, pp. 16-25, Department of Artificial Intelligence, Edinburgh University, March 2001.

[61] AMZI: **http://www.amzi.com/ExpertSystemsInProlog**/, 2009.

[62] Swedish Institute of Computer Science: **SICStus Prolog User's Manual**, May 2007.

[63] Almeida, P.: **Metodologias para Inicialização dos Modelos de Aluno e da Base de Conhecimento de um Sistema de Seguimento de Alunos**, Technical Report, University of Minho, Braga, Portugal, January 2009.

[64] Agrawal R., Imielinski T., Swami A.: **Mining association rules between sets of items in large databases**, Buneman and Jajodia, editors, in Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 207–216, Washington, D.C., USA, 1993.

[65] Agrawal R., Srikant R.: **Fast algorithms for mining association rules**, Bocca, Jarke, and Zaniolo, editors, in Procedures of the 20th International Conference on Very Large Data Bases, VLDB, pp. 487–499, Morgan Kaufmann Publishers, Los Altos, California, USA, 1994.

[66] Weka: **http://www.cs.waikato.ac.nz/ml/weka**/, 2008.

[67] Davies, D., Fernandes, J.V., Soares, J.C., Lourenço, L., Costa, L., Vilas-Boas, M.A., Vilhena, M.C., Oliveira, M.T., Dias, M., Silva, P., Marques, R. e Lima, R.: **As Escolas e as Famílias em Portugal: realidade e perspectivas**, Lisboa, Livros Horizonte, 1989.

[68] Coach, B. and Siegle, D.: **A comparison of high achievers' and low achievers' attitudes, perceptions, and motivations**, Academic Exchange Quarterly, June, 22, 2001.

[69] Aamodt, A. and Plaza, E.: **Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches**, AI Communications, Vol. 7 Nr. 1, pp 39-59, March 1994.

[70] Almeida P., Novais P., Neves J.: **Prediction Tools for Student Learning Assessment in Professional Schools**, in Proceedings of the ESM 2008 - The 22nd annual European Simulation and Modelling Conference, Le Havre, France, October, ISBN 978-90-77381-44-1, pp 121-129, 2008.

[71] Kruck, S. and Lending, D.: **Predicting Academic Performance in an Introductory College-Level IS Course**, Information Technology, Learning, and Performance Journal, Vol. 21, No. 2, Fall 2003.

[72] Forgy C.: Rete: **A Fast Algorithm for the Many Patterns/Many Objects Match Problem**. Artificial Intelligence Journal, 19(1):17– 37, 1982.