AMI-RIA: Real-Time Teacher Assistance Tools for an Ambient Intelligence Classroom

Georgios Mathioudakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Advisor: Prof. *Constantine Stephanidis*

This work has been supported by the Institute of Computer Science of the Foundation for Research and Technology-Hellas (FORTH)

University of Crete
Computer Science Department

AMI-RIA: Real-Time Teacher Assistance Tools for an Ambient Intelligence Classroom

Thesis submitted by
Georgios Mathioudakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: Georgios Mathioudakis

Committee approvals:

Constantine Stephanidis
Professor, Thesis Supervisor

Dimitris Plexousakis
Professor, Committee Member

Dimitris Grammenos
Principal Researcher, Committee Member

Departmental approval:

Angelos Bilas
Professor, Director of Graduate Studies

Heraklion, October 2012

AMI-RIA: Real-Time Teacher Assistance Tools for an Ambient Intelligence Classroom

Georgios Mathioudakis
Master's Thesis
Computer Science Department, University of Crete

Abstract

Ambient Intelligence (AMI) represents a vision of the future where people are surrounded by technological means that are sensitive and responsive to their behaviors. The promising potentials of AMI in education led to the introduction of the notion Intelligent Classroom. An Intelligent Classroom is an environment where conventional classroom activities are enhanced with the use of pervasive and mobile computing, artificial intelligence and multimedia. However, the majority of current research approaches towards the Intelligent Classroom address issues focusing on the learner's perspective, with much less attention so far to the role of the instructor.

The AMI-RIA system proposed in this thesis aims to support teachers during the educational process in the classroom, by providing facilities for observing in real time the students' activities and mechanisms for identifying possible related problems and issues which need to be addressed at an individual or classroom level. This information is used to make the teacher aware about the students' status, therefore offering the possibility to intervene, providing help or adapting the teaching process.

In particular, the developed system provides an infrastructure of intelligent agents for monitoring the students in an unobtrusive way and exploits the advantages of ontologies to model their activities and apply reasoning on them. An application for the teacher has been developed to present a real-time live

view of the classroom and provide statistics and graphs about the students' progress and performance. Finally, a set of tools are incorporated in the system, enhancing the typical procedures that can be found in conventional classrooms.

AMI-RIA: Πραγματικού Χρόνου Εργαλεία Υποστήριξης του Καθηγητή σε μία Σχολική Τάξη Διάχυτης Νοημοσύνης

Γεώργιος Μαθιουδάκης
Μεταπτυχιακή Εργασία
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Περίληψη

Η διάχυτη νοημοσύνη (AMI) αποτελεί ένα όραμα για το μέλλον στο οποίο οι άνθρωποι περιβάλλονται από τεχνολογικά μέσα που είναι ευαίσθητα και ανταποκρίνονται στις συμπεριφορές τους. Οι πολλά υποσχόμενες δυνατότητες της διάχυτης νοημοσύνης στην εκπαίδευση οδήγησαν στην εισαγωγή της έννοιας της έξυπνης τάξης. Η έξυπνη τάξη αποτελεί ένα περιβάλλον στο οποίο οι συμβατικές δραστηριότητες της τάξης ενισχύονται με τη χρήση διεισδυτικών και φορητών υπολογιστών, τεχνητής νοημοσύνης και πολυμέσων. Ωστόσο, η πλειοψηφία της τρέχουσας έρευνας προς την έξυπνη τάξη αντιμετωπίζει προβλήματα εστιάζοντας στην πλευρά του μαθητή, δίνοντας μέχρι τώρα πολύ μικρότερη προσοχή στον ρόλο του καθηγητή.

Το σύστημα AMI-RIA που προτείνεται στην παρούσα διατριβή, έχει ως στόχο να υποστηρίξει τους καθηγητές κατά τη διάρκεια της εκπαιδευτικής διαδικασίας στην τάξη, παρέχοντας την υποδομή για την παρατήρηση σε πραγματικό χρόνο των δραστηριοτήτων των μαθητών και τους μηχανισμούς για την αναγνώριση πιθανών σχετικών προβλημάτων που χρειάζεται να αντιμετωπιστούν μεμονωμένα ή στο επίπεδο της τάξης. Αυτή η πληροφορία χρησιμοποιείται για να κρατάει τον καθηγητή ενήμερο σχετικά με την κατάσταση των μαθητών και επομένως του δίνει τη δυνατότητα να παρέμβει, παρέχοντας βοήθεια ή προσαρμόζοντας την εκπαιδευτική διαδικασία.

Ειδικότερα, το σύστημα που υλοποιήθηκε παρέχει μια υποδομή έξυπνων πρακτόρων για την παρακολούθηση των μαθητών με ένα διακριτικό τρόπο και εκμεταλλεύεται τα πλεονεκτήματα των οντολογιών για να μοντελοποιήσει τις δραστηριότητες τους και να εφαρμόσει συλλογιστική πάνω σε αυτές. Μια εφαρμογή για τον καθηγητή κατασκευάστηκε για να προβάλει μια πραγματικού χρόνου απεικόνιση της τάξης και να παρέχει στατιστικά και γραφήματα σχετικά με την πρόοδο και απόδοση των μαθητών. Τέλος, ένα σύνολο από εργαλεία ενσωματώθηκαν στο σύστημα, ενισχύοντας τυπικές διαδικασίες που μπορούν να βρεθούν σε συμβατικές σχολικές τάξεις.

Table of Contents

TABLE OF CONTENTS ..... **IX**

LIST OF TABLES ..... **XII**

LIST OF FIGURES ..... **XIII**

INTRODUCTION **1**

1.1 AMBIENT INTELLIGENCE 1

1.2 TOWARDS AMBIENT INTELLIGENCE IN EDUCATION 2

1.3 MOTIVATION FOR A REAL-TIME TEACHER ASSISTANCE TOOL 3

1.4 OVERVIEW OF THE SYSTEM 4

1.5 A REAL-WORLD SCENARIO 5

1.6 ORGANIZATION OF THIS THESIS 7

RELATED WORK **8**

2.1 SMART CLASSROOM 8

2.2 STUDENT MONITORING IN THE CLASSROOM 11

2.3 STUDENT MONITORING IN E-LEARNING ENVIRONMENTS 16

2.4 CONTRIBUTION OF THE AMI-RIA SYSTEM 20

SYSTEM IMPLEMENTATION **22**

3.1 ARCHITECTURE 22

3.2 DESK MONITOR ARCHITECTURE 23

3.2.1 Data collection from the students' desks *23*

3.2.2 Desk Monitor components *25*

3.3 TEACHER ASSISTANT ARCHITECTURE OVERVIEW 27

3.4 DATA REPRESENTATION AND REASONING 29

3.4.1 RDF *29*

3.4.2 RDF Schema *30*

3.4.3 Semweb.NET *30*

3.5 KEYWORDS MATCHING ENGINE 31

STUDENT MONITORING **35**

4.1 IDENTIFYING THE ACTIVITIES 36

4.2 STUDENT ACTIVITIES REASONING 38

4.2.1 Identify an off-task student *39*

4.2.2 Identify an inactive student *40*
4.2.3 Identify a student that requires attention during an exercise *42*
Student disengagement 42
Problem on exercise completion 43
Problem on exercise skipping 44
4.2.4 Extendibility *44*

TEACHER ASSISTANT 46

5.1 DATA FLOW 46
5.2 DATA MANAGEMENT 47
5.2.1 Taxonomies *48*
5.3 THE DESIGN 51
5.4 TEACHER AUTHORIZATION 53
5.5 THE CLASSROOM LIVE VIEW 55
5.5.1 Classroom layout *56*
5.5.2 The student card *56*
Student states 56
Tasks 58
The Action Logger 59
Exercise related information 60
5.5.3 Filtering mechanism *62*
5.5.4 Attendance *63*
5.6 TESTS AND ASSIGNMENTS 64
5.6.1 Assignments *64*
5.6.2 Tests *65*
5.7 THE SHORT-TERM REASONER 69
5.8 CLASSROOM ACTIVITIES REASONER 72
5.8.1 The Classroom Activities Reasoner rules *72*
5.8.2 The classroom activities reasoner notifications *73*
5.9 THE STATISTICS MODULE 74
5.9.1 Threading & Caching *79*
5.10 THE CLASSROOM SCHEDULE 80

EVALUATION 82

CONCLUSION AND FUTURE WORK 92

5.11 SUMMARIZING 92
5.12 FUTURE WORK 93

BIBLIOGRAPHY **95**

List of Tables

Table 1: Processing time for the keywords indexing procedure 34
Table 2: Number of supported desks proportionally to screen resolution 51
Table 3: The usability heuristics 84
Table 4: Severity ratings 84
Table 5: Complete list of usability issues discovered during the heuristic evaluation process 91

List of Figures

Figure 1: The Student Tracking User Interface of MiGen 12
Figure 2: Retina Instructor View in Browse Mode 14
Figure 3: I-Minds - The teacher agent interface 16
Figure 4: Chernoff faces representing students 17
Figure 5: The classroom view with Chernoff faces 17
Figure 6: Course-Vis Cognitive Matrix 19
Figure 7: The AMI-RIA system architecture 23
Figure 8: Recognition of book pages and their localization on the SESIL system 24
Figure 9: a) The augmented student desk b) The SMARTboard 24
Figure 10: Desk Monitor architecture 25
Figure 11: The Teacher Assistant architecture 27
Figure 12: Example of Notation3 rule 31
Figure 13: The “In context” rule 38
Figure 14: Taxonomy example 49
Figure 15: A full representation of the classroom ontology 50
Figure 16: User interface on a 1024 X 768 tablet device (scaled) 52
Figure 17: User-interface in 1280 X 1024 screen resolution (scaled) 52
Figure 18: Application's main menu with descriptive labels (scaled) 53
Figure 19: The footer bar of the teacher application (scaled) 53
Figure 20: MagTek MagneSafe Mini card reader 54
Figure 21: The front side of the magnetic card for the teacher 54
Figure 22: The login screen of the Teacher Assistant 55
Figure 24: Student in inactive state 57
Figure 23: Student in active state 57
Figure 25: Student in out-of-context state 57
Figure 26: The study task 58
Figure 27: The exercise task 58
Figure 28: The image gallery task 59

Figure 29: Action logger pop-up window (scaled) 60
Figure 30: Exercise progress pop-up window (scaled) 61
Figure 31: Exercise details and student's statistics pop-up window 62
Figure 32: Filtering mechanism in sidebar (scaled) 63
Figure 33: The attendance log (scaled) 64
Figure 34: The homework assignments window 65
Figure 35: The exercise file structure 66
Figure 37: The label indicating the test process 67
Figure 36: The tests window 67
Figure 38: Examination results pop-up window 68
Figure 39: The test reports panel 69
Figure 40: A short term reasoner notification 70
Figure 41: The STR preferences window (scaled) 71
Figure 42: Notifications generated by Classroom activities reasoner 74
Figure 43: Statistics panel – All course/ My courses selection 75
Figure 44: Statistics panel - Time period selection 76
Figure 45: Student selection mechanism incorporated in left sidebar 76
Figure 46: Classroom statistics (scaled) 77
Figure 47: Classroom statistics and bar charts (scaled) 78
Figure 48: Student statistics and bar charts (scaled) 79
Figure 50: Weekly schedule view 81

Introduction

1.1 Ambient Intelligence

The continuous evolution of ICT (Information and Communication Technologies) has a drastic impact on the way people interact with computers and electronic devices. Nowadays, people are hooked on connectivity, and enjoy access to information anytime and anywhere. This evolution is leading to the development of novel concepts and tools to provide content-rich invisible computing.

Ambient intelligence (AmI) is a vision of the future of computing, electronics and telecommunications that refers to electronic environments that are sensitive and responsive to the presence of people. In an AmI environment, devices work in a natural way to support people in carrying out their everyday life activities. As these devices grow smaller, more connected and more integrated into the environment, technology disappears into the surroundings until only the user interface remains perceivable by users when necessary. Ambient intelligence research builds upon advances in sensors and sensor networks, pervasive computing, human computer interaction and artificial intelligence. Because these contributing fields have experienced a tremendous growth in the last few years, AmI research has strengthened and expanded.

There are many settings in which AmI can greatly influence everyday life. Some of the emerging applications include home automation, communication, entertainment, work and education. All these applications share as a common feature the purpose of responding to the users' needs and preferences. This thesis investigates the potential of AmI in education and the way AmI applications can assist teachers in real classrooms.

1.2 Towards ambient intelligence in education

Probably the most popular realization of ambient intelligence is the "Smart Home" [1]. The majority of electrical appliances in a house can be easily extended with sensors and programmed to gather information from the environment. This information can be exploited by the devices individually or in combination in order to act accordingly, responding to the user needs and preferences.

AmI researchers had a vision that the installation of such "smart" technologies in learning environments and classrooms could have a significant impact on the learning process. A team at the Georgia Institute of Technology developed and proposed one of the first "Smart Classroom" environments. The "Classroom 2000 project" [2] aimed to facilitate content to lecturers and students by capturing the lectures for later use without disrupting the usual classroom activities. Furthermore, the students could write notes in the slides during the presentation. After the lecture, these notes could be converted into HTML and published on the web. The evaluation of the system showed a positive reaction to this prototype. Students found the web-based review notes interesting and useful for examining their own notes and the teacher's notes.

Another smart classroom prototype proposed by Shi et al. [3] at Tsinghua University uses an interactive whiteboard and allows lecturers to write notes directly on the board with a digital pen, edit the text, make drawings and other relevant tasks. A second board is used for supporting remote students. The teacher and students in the classroom can view and interact with remote students through this board. A network of cameras is used to understand specific situations and adapt the service to the current context. When the teacher is writing on the whiteboard, the cameras zoom in to capture the notes, but when the teacher is talking to the classroom, they zoom out to capture a larger view. The lecture, recorded as hypermedia courseware, is made available for playback after the class. The evaluation of the system with several teachers

received positive comments mainly for the features regarding the remote students.

The evolution of internet and World Wide Web pointed out a more information-centric aspect of AmI in education. Towards this end, several systems were implemented to support personalized learning. In [4] authors describe a system that encapsulates a content classification mechanism, which provides the necessary content-related information for data mining procedures. User profiles are created to store the essential static and dynamic characteristics of the students. These profiles contain data about preferred exercise types, preferred multimedia types, preferred exercise level and more. During the learning process, a sophisticated filtering mechanism is used for providing personalized material based on the user needs and preferences, available through the student profile.

1.3 Motivation for a real-time teacher assistance tool

Current research studies focused mostly on supporting the student, either by extending technologically the classroom to support activities like note-taking, controlling the lights or shades, etc. or by providing personalized content. Nevertheless, can the teacher benefit from such "smart" environments and in what way? This thesis aims to answer this question and investigate the need for a system to support the teacher in the classroom.

During the educational activities in a classroom, the instructor is responsible for the teaching process but, at the same time, he/she is responsible to observe the students and intervene if a student needs help regarding the current lesson. In the case that the number of students in the classroom is relatively small, this is an easy task for the teacher to accomplish. However, in a more crowded classroom the task of observing the students becomes fairly heavy and time-consuming. This generates the need of a system to observe the students' activities, produce valuable insights on them and provide the teacher all the necessary information at run-time. In this way, the teacher could

distinguish the students that have problems and intervene by providing help. Furthermore, based on the students' performance and progress the instructor could easily locate learning gaps and adapt the teaching process and the educational material accordingly.

Additionally, the typical procedures appearing in conventional classrooms could become difficult for the teacher to manage when a large number of students are involved. Such procedures include homework assignments, the tests, attendance records, students' performance statistics and other tasks. In an intelligent classroom environment, technological means can be exploited to offer digital versions of these procedures, thus making them easily manageable by the teachers or the school administrators.

1.4 Overview of the system

To achieve the above objectives, this thesis introduces a system named AMI-RIA (Ambient Intelligence Real-time Instructor Assistant) for supporting the instructor in the smart classroom environment. The AMI-RIA system aims to enhance the role of the teacher by providing facilities for monitoring the students' activities during lessons in an unobtrusive way. The collected data about the activities are processed by an inference mechanism in order to generate further insights regarding the students. The teacher is notified in real-time about the activities occurring in the classroom and any possible problems. In this way, the teacher does not need to go through the desks in order to observe the students; any information needed is always available at his tablet device or personal computer at the teacher's desk. Therefore, he/she can focus on the teaching process leaving the monitoring task for the system.

The system consists of two major components, the *Desk Monitor* and the *Teacher Assistant.* The *Desk Monitor* is an agent-type module responsible for collecting all the data generated by the students during the learning process. This information is stored in an internal data-model, using the Resource Description Framework (RDF) and semantic taxonomies. A real-time, rule-

based reasoner is used to exploit these semantic data by generating inferences. The inferences indicate possible problems of the students like disengagement from the lesson, bad performance, etc. Therefore, the system assists the teacher in observing the students' activities and taking decisions during lessons, by providing run-time information, which would be hardly available otherwise.

The data and inferences are transmitted in real-time to the teacher desk, where a system named *Teacher Assistant* is deployed, to present to the teacher an overview of the classroom and all the information originating from the student desks. Furthermore, a second reasoning service is utilized at a classroom level to identify any patterns that the teacher should be aware of. Therefore, he/she can provide help to the classroom or further explain the problem, thus adapting the teaching process to the students' needs.

The *Teacher Assistant* is also used to provide to the teacher several tools that are useful for the classroom management. Such tools are the attendance record, the homework assignments, the testing process and more. Finally, a statistics module is provided that generates graphs and statistics about the classroom's and individuals' progress and performance.

1.5 A real-world scenario

This section describes a real-world scenario of a classroom using the teacher assistance tools developed in the context of this thesis. Let us imagine a typical day at a 3rd grade classroom in a primary school, in a class consisting of twenty-five students.

A typical school day begins, after the ringing of the bell, the students are gathering in the classroom. Using their personal notebooks that are equipped with RFID tags, the students gradually log in their desks. The first course of the day starts and the teacher opens the attendance record to view any students who are not present in the classroom. The course is about physics and in

particular gravity. The last time the teacher had assigned homework to the students, thus now the desks ask them to submit their homework.

The students open gradually their books to the chapter on gravity as the teacher indicated. Five minutes later, he observes on the classroom overview panel of the system that two students appear to be “off-task”. In fact, one of them is sleeping on his desk and the other opened the book of English grammar by mistake. The student desks appear on the user-interface according to their position in the classroom, thus the teacher can locate the two students easily and help them navigate to the right chapter of the Physics book.

Before the end of the lecture, the teacher encourages the students to start a multiple-choice exercise on the topic discussed. He observes the students as they start the exercise. At any time, he can view the current score of each student and what questions he/she has answered correctly or not. Some minutes later, the system notifies the teacher about three students that started the exercise but in the meantime, they got distracted and started chatting with each other. He locates these students in the classroom and motivates them to continue working on their task again.

During the next hour, the English teacher starts a scheduled test on English grammar. The corresponding exercises for the test appear automatically on the students’ desks. About thirty minutes later, the students gradually start to complete the test. The teacher focus on three students who completed the test but the system warned him that they had some problems. He looks into their answers and decides to help them by providing some extra material. Before the end of the lesson, the teacher assigns an exercise from the book as homework until the next lesson of English.

Some day at the ending of the semester, the teacher goes through the students’ statistics in order to advise their parents. The statistics module generates graphs on the students’ progress and performance based on exercises and tests on the topics covered in the lessons. As the teacher review the statistics of John, a student in his class, he observes that he encountered

difficulties in two topics of the physics lesson. He decides to provide some extra material to John and advise his parents to focus on these topics.

The above scenario points out the potential of using such a supporting system in the real classroom environment, especially if the number of students is large (e.g., twenty or more).

1.6 Organization of this thesis

The rest of this thesis is structured as follows:

- Chapter 2 presents an overview of related work
- Chapter 3 describes the design and architecture of the developed system and presents the technologies that were used during the development
- Chapter 4 introduces the first of the two major components of the system, the “Desk monitor”
- Chapter 5 introduces the second major component of the system, the “Teacher assistant”
- Chapter 6 presents the evaluation of the system and examines the related results
- Chapter 7 summarizes the conclusions of this thesis and outlines steps for future work.

Related Work

2.1 Smart Classroom

The potential of AMI in education has inspired several research teams to work on the Smart classroom paradigm. The research projects that aim to realize an intelligent classroom describe both hardware and software components and the way they integrate on the environment in order to achieve the pervasive nature.

Ramadan et al in [5] describe an intelligent classroom called *iClass* that aims to realize the Ami vision in Education in universities and schools. *iClass* consists of a large number of embedded sensors, actuators, processors and a heterogeneous network. In particular, *iClass* utilize sensors for measuring internal and external light, internal and external temperature and humidity. The data collected by the sensors are used to take controls of classroom artifacts, like dimmable spot-lights, window blinds and air conditioning. The sensors and actuators are hidden in the classroom so that the users are completely unaware of this infrastructure. Furthermore, the *iClass* system introduced an RFID and a speech recognition system. With the use of RFID technology, the authors aimed at recognizing the presence of teacher and students in the classroom. The system used this data to load the appropriate educational content and to keep organizational information such as the attendance record. Finally, the authors utilized a speech recognition system for controlling the electric devices of the classroom. They benefited from this component by providing a set of commands for lights, curtains and air-conditioning. Although being an interesting system from an Ambient Intelligence point of view, *iClass* appear to be more targeted to the environmental control in the classroom rather than to support learning and teaching.

In [4], the authors discuss an education-centric approach towards ambient intelligence in the classroom and propose an integrated architecture for

pervasive computing environments, named *ClassMATE* (Classroom Multiplatform Augmented Technology Environment). ClassMATE monitors the ambient environment and makes context-aware decisions in order to assist the student in conducting learning activities, by simplifying everyday tasks and providing personalized content according to individual needs. ClassMATE aims to offer an open ubiquitous computing framework suitable for a school environment. To achieve this, the authors introduced two modules namely, the Ambient Environment Manager and in the Content Personalization Manager. The first of them is responsible to monitor the environment and take actions on the devices appearing in the classroom, whereas the latter is responsible for the delivery of personalized educational content based on the current needs of the individual learner. In order for an assessment of the educational process in smart classroom, a number of educational tools were developed, namely the ClassBook, the Multimedia application, the Multiple-Choice Exercise and the Hints application. In particular, the ClassBook application is the electronic version of a physical book displaying the current page, images and exercises. The Multimedia application displays the multimedia content relevant to a topic. The Multiple choice exercise application is the online representation of multiple-choice exercises and last, the hints application launches when the student asks for help about a specific exercise. *ClassMATE* collaborates closely with the PUPIL [6] system that has been developed and deployed in the smart classroom devices and provides the educational front-end of the system.

O'Driscoll et al in [7] present a Context Aware Smart Classroom (CASC) that responds to lecturers and student groups based on preset policies and the lecture timetable. The purpose of developing the CASC system was to leverage existing technologies such as the personal devices of students and lecturers as Bluetooth tags to permit the existing communications infrastructure provided by WLAN, LAN and email to enhance the students experience in classroom. A typical scenario of the system involves identifying a lecturer walking into the classroom. The system retrieves the presentation notes for the particular lecture and displays them on a projector and additionally it distributes the notes

to students wirelessly. In general, the system is designed to react to changes in the environment according to rules preset in the system. A rule-based algorithm is used to check information stored in local database and make decisions based on the system context and the preset policies. Three main modules constitute the system architecture, the smart classroom manager, the policy manager and the context manager. The smart classroom manager is the core of the CASC system. It is designed to provide adaptive behavior based on the set of system rules. The smart classroom manager collects the policy settings of the users and the current context by connecting to an appropriate database and is responsible for transferring material to students according to student specific policies. The policy manager keeps a set of settings and preferences for lecturers, students, rooms and notes. The context manager module is responsible for collecting real-time data and store the information into a database for use by the smart classroom manager. The context manager relies on a Bluetooth monitoring daemon on a main computer for communicating with user devices. The evaluation of the system showed that the use of the Bluetooth technology for identifying lecturers and students raised concerns about the potential scalability of this identification technique.

In [8], the authors propose a real-time assistance environment named *S-CRETA* (Smart Classroom Real-Time Assistance), which aims to recognize any occurring activity in a smart classroom. The system exploits the advantages of semantic ontologies in order to model the context and provide reasoning on high-level activities recognition. A typical scenario for the S-CRETA system describes a student presentation. When a student is about to start the lecture the system recognizes the activity and automatically lowers the lights and bring the presentation file on students' and teachers' devices. Furthermore, the authors utilize machine-learning algorithms in order to recognize student activities. A complete cycle of the system includes collecting data from sensors and store them in an ontology, use SWRL rules for a first level reasoning, pass the simple events to an activity recognition system, find the current activity by loading the cases in the activity monitor, write the new case in the case base and

finally take actions depending on the current activity. *S-CRETA* introduces novel technologies in the AmI environment such as semantic web and machine learning. However, the system is focused on the control of the devices appearing in the classroom rather than assisting the learning and teaching processes.

2.2 Student monitoring in the classroom

The use of ICT in real classroom environments is becoming widespread. These technologies tend to integrate seamlessly, realizing the pervasive computing paradigm. Researchers now have the potential to take advantage of this technological equipment inside the classrooms to enhance the learning and teaching process. Towards this objective, intelligent systems have been developed to monitor the students' activities and assist the teachers by reporting valuable insights.

In [9], the authors have designed and developed an intelligent environment named *MiGen*, to support 11-14 year-old students in their learning of algebraic generalisation. The system aims to assist teachers in developing teaching strategies, informing them of students' progress, the appearance of misconceptions, students who may be disengaged from the task and students who may encounter difficulties. As the authors state, this will allow teachers to support learning in a personalized way. The system introduces a number of landmarks that define students' activities during the learning process. Landmarks can be either task-dependent like "correct global exp" or task-independent like "inactive". *MiGen* targets at visualizing students' progress through constructionist learning tasks and notifying teachers of students' attainment of specific landmarks as they are undertaking their constructions. The system consists of three main modules namely, the eXpresser, the eGeneralizer and the teacher assistance tools. The eXpresser module is a mathematical micro-world which supports the students in undertaking mathematics generalization tasks. The eGeneraliser is a suite of intelligent components, which take as their input information from the eXpresser as

students are undertaking tasks, as well as information in the MiGen database relating to students and to tasks. These components make the basic inferencing and generate real-time feedback for students. A suite of teacher assistance tools aims to help the teacher in reporting students' activities and progress. Figure 1 presents the student tracking user-interface showing the number of times each landmark has occurred for each student during the current task. A green marked landmark of a number indicates that the actions of the student are in line with what would be expected from constructive interaction with the system. A red marked landmark is regarded as an obstruction to constructive use of the system e.g. off task or inactive. Finally, a yellow landmark indicates that the student is showing some understanding of how to use the system but it is not quite on the right track. Although *MiGen* introduces some interesting features regarding the students monitoring and teacher assistance, its use is limited to the algebraic generalization and therefore it cannot be used as a general assisting tool in a classroom environment. Furthermore, the interface of the system is rather confusing and full of information making it hard for the teacher to locate a specific student or landmark, thus making it difficult to extract any information.

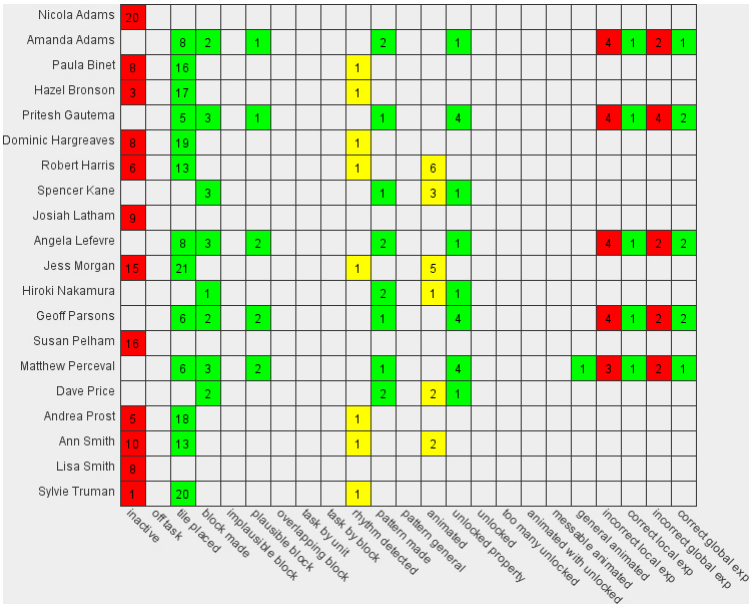


Figure 1: The Student Tracking User Interface of MiGen

Murphy et al in [10] propose a tool named *Retina,* which collects information about students' programming activities, and then provides informative reports to both students and instructors based on the aggregation of that data. *Retina* aims to assist the instructors by keeping them informed about the problems their students are having, thus making them capable to address those in lecture and tailor future assignments accordingly. At the same time, *Retina* provides real-time recommendations to students in order to help them address some of the errors they make. Finally, *Retina* logs past information about students' activities and makes them available to the instructor, who is responsible to identify the types of problems the students had with assignments in previous semesters and try to address them. Four main modules form the *Retina* system, the data collector, the instructor view, the student view and the recommendation system. *Retina* records students' compilation attempts and compiler errors so that they can be stored in a central database and later mined and analyzed. Once the data has been collected, instructors can access the collected data through the Retina Instructor View, which is implemented as a standalone Java application that runs locally on the instructor's system.

As presented in Figure 2 the instructor can select a student and an assignment for either the current semester or a past one, and then see a list of all of the student's compilation and runtime errors, as well as aggregate data about the total number of compilations, the total number of compilation errors, and the most common compilation error. Students can access their own Retina information via JSP pages on a web server. A student who logs into the Retina student view can see information about his activities and his status related to the class. Finally, Retina proposes a feature that has the ability to produce immediate, real-time recommendations that can proactively be sent to students based on their observed programming activities. The authors admit that there are privacy concerns when it comes to collecting information and sharing it with others. However, the evaluation of the tool showed that the students who participated in the process found it acceptable.

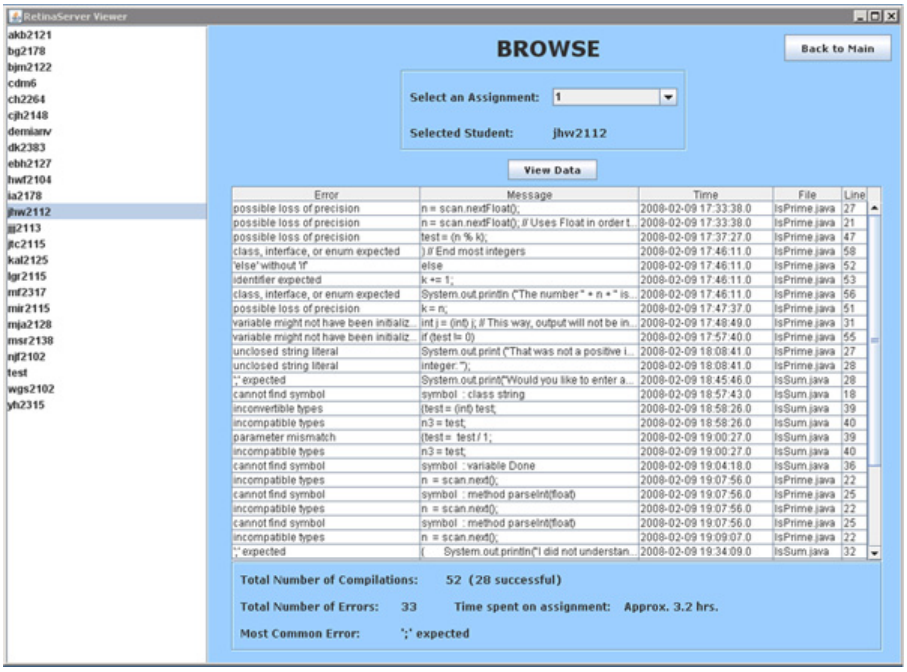


Figure 2: Retina Instructor View in Browse Mode

Although *Retina* makes a great contribution towards monitoring systems its use is limited to the programming courses as it depends on the compilation errors and times. Additionally, the implementation of the instructor view is partially helpful for the instructor during the learning process, as he needs to search for a specific student in order to discover any problems.

Another study focusing on the programming environment is [11]. Rodrigo et al. propose a system focused on detecting student frustration, at a coarse-grained level, using measures distilled from student behavior within a learning environment for introductory programming. Researchers recognize that frustration is potentially a mediator for student disengagement and eventually attrition. They endeavor to detect frustration in order to help the instructor to intervene in ways that will help students persevere. The monitored data includes the computer number, the time stamp of the compilation, success or failure of the compilation, error message, name of the compiled file, and source

code. Using Weka[1], the authors generated a linear regression detector of average student frustration across five lab exercises. Evaluations results showed that the system could be sufficiently fine-grained to drive meaningful instructional interventions after five sessions. However, the authors focus on the affective states identification and do not provide any details on how the results of the monitoring will be presented to the teacher and in what way this information would assist him/her.

In [12], the authors argue that teachers working in robotic classes have problems in keeping track of student activities. As they claim, the challenge for instructors is to know when to intervene and how to intervene. The design of this system is based on the LeJOS[2] programming platform for Lego Mindstorms[3]. In order to monitor students' activities, the authors propose two agent modules for data collection. One of them is embodied into the robot and the other inhabits the programming environments, monitoring the students' programming activities. However, in this work the user-interface for the teacher application is not presented and cannot be evaluated.

In [13], [14], [15] and [16], an infrastructure is described to help the instructor to manage large or distance classrooms and support students' participation and collaboration. They propose a system named I-MINDS that consist of a group of intelligent agents, student agents and teacher agents. The student agents include a mechanism that tracks the activities and the progress of the student. This tracking mechanism identifies any problematic situations and acts accordingly assisting the student, for example when a student is inactive a sound may play to alert him/her to concentrate on class. The information tracked by student agents is transmitted to the teacher agent in order to let the teacher know who needs help or what concepts need emphasize in class. Therefore, the teacher knows who is doing what, who is asking

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] http://lejos.sourceforge.net/

[3] http://mindstorms.lego.com/en-us/Default.aspx

questions, who is not participating etc. I-MINDS constitutes an extended work on learners activity monitoring; however, it is targeted on the social activities of the students rather than the learning activities. The monitoring process is based mostly on the questions the students ask and the responses they give to other students or groups. Figure 3 presents the teacher agent interface showing the students' questions, their classification and scores.

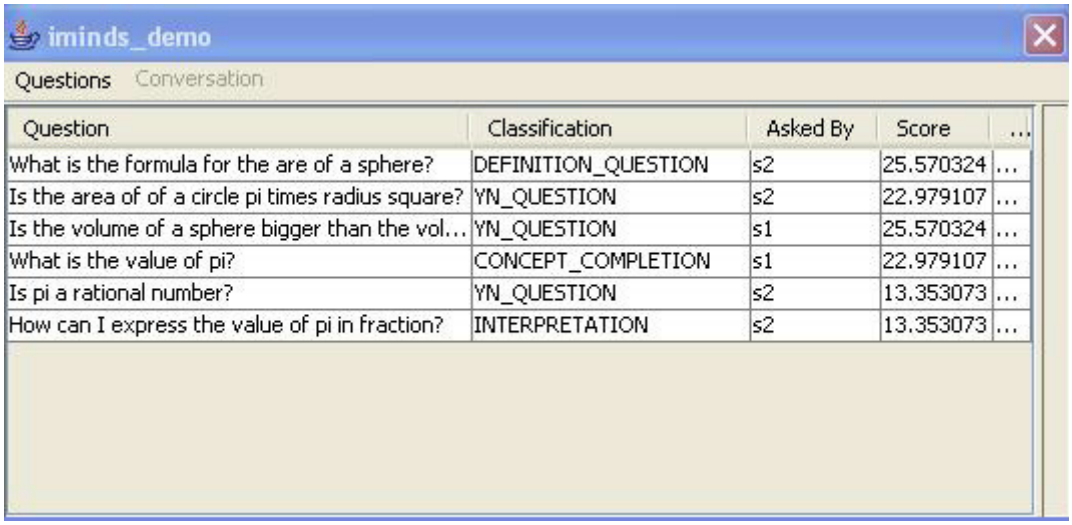


Figure 3: I-Minds - The teacher agent interface

2.3 Student monitoring in e-learning environments

The smart classroom paradigm usually refers to real classroom environments. However, a fair number of systems have been created for teacher assistance in web-based environments. Some of them involve innovative methods of visualising the students' activities, therefore they worth mentioning in this section.

In [17] and [18], the authors propose a web based environment to help the teacher visualize a virtual classroom and interact with it. The monitoring process is achieved thanks to the traces left by students during their activities within a web-based learning environment. An activity is defined as a set of

actions with a specific goal, e.g., an exercise. In order to visualize these traces, the authors introduce a way of representing the students as Chernoff faces [19].

The Chernoff faces characteristics dynamically evolve in time, according to the students activities. For example, if a student had no activity during the last minutes, the eyes of his Chernoff face are gradually closing. Figure 4 illustrates all the available Chernoff faces and characteristics. The color of the face changes according to the time spent by the student in an activity; the longer a student works on it, the darker the face becomes. Additionally, the system represents the pedagogical activities as bubbles, which grow or shrink proportionally to the number of students doing them. As presented in Figure 5, this way the students are divided in groups based on their activities. The background color of a bubble corresponds to the relative delay of the students implied in this activity compared to the average time required. The paths between the bubbles indicate that a student completed an activity and started another one. The path's width indicates the number of times that the path was taken by the students.

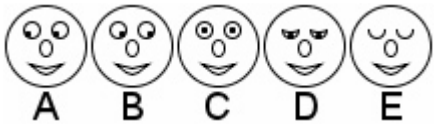


Figure 4: Chernoff faces representing students

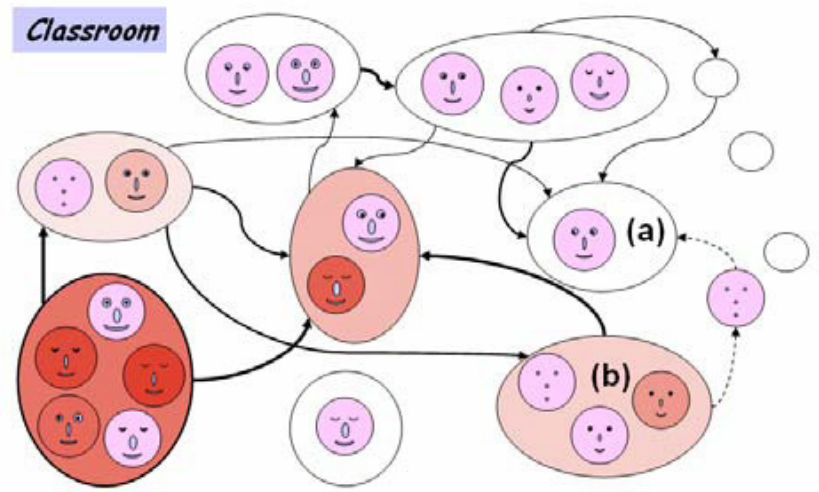


Figure 5: The classroom view with Chernoff faces

This work makes a strong contribution on the topic of classroom visualization for teacher assistance. However, the proposed system could only fit in a real classroom only if the students operate conventional computers. Furthermore, the information the teacher gets is limited to the activities duration.

Mazza and Dimitrova in [20] and [21], present a tool called *CourseVis* that uses student-tracking data collected by a Course Management System (CMS) to generate graphical representations that can be used by instructors to gain an understanding of what is happening in distance learning classes. The CMS stores in its internal database large log files about the students' activities in the courses. Furthermore, it has a built-in student-monitoring feature that enables the instructors to view some statistical data, such as the number of accesses made by the students to each resource, the history of pages visited, and the number of hits for every day of the course. CourseVis also proposes an approach to represent graphically student-tracking data generated by the CMS. The data tracked by the CMS are exported in XML format and then need to be imported in the *CourseVis* system, thus real-time graphics are not supported. Based on that data, the system generates graphics that visualise social, cognitive and behavioral aspects of the learners. Towards this end, several plots are presented namely, the discussion plot, the cognitive matrix plot, the student access plot and the behavior graph. The discussion plot shows the discussion threads between the learners. The cognitive matrix visualise the learners' performance on quizzes. The student access plot represents the learners' attendance to the online course and finally the behaviour graph represents the learners' overall behaviour meaning his/her progress, access to content pages, messages and assignment submission. Figure 6 presents the Cognitive Matrix that visualizes the students' performance on quizzes. The way the data is presented to the instructor is rather complicated and it could become difficult to extract any information from it.

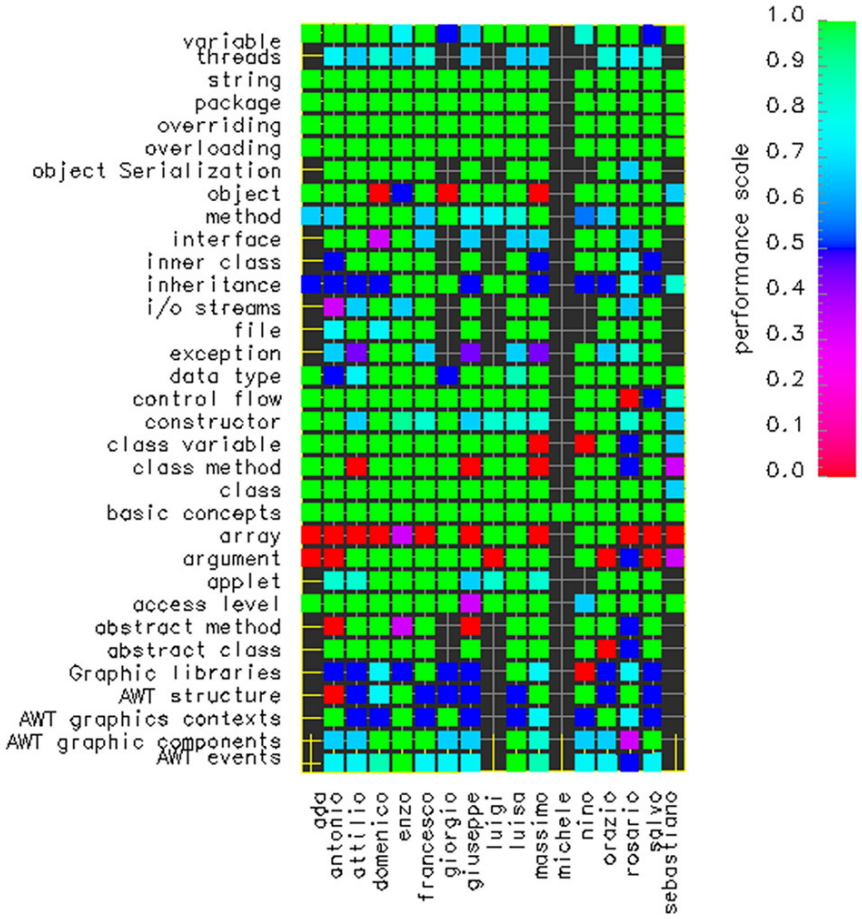


Figure 6: Course-Vis Cognitive Matrix

Another monitoring system based on CMS is [22]. The authors present an intelligent agent system to support teachers in supervising learners and activities in LAMS (Learning Activity Management System). The proposed system is able to notify the teacher for common problems about participation and contribution of students and groups during the activities. In order to produce valuable results the teacher must predefine his expectations for the attendance and the contribution of the learners as parameter values for each activity. These values will be used by the system for the determination of learners that have stretched the time limits. The activity parameters involve minimum/maximum time required to spend on each activity, minimum/maximum contribution in each collaborative activity and minimum/maximum score in each evaluative activity. These values are used by

the system for the production of alert messages and for the evaluation of users and activities related to the lesson.

2.4 Contribution of the AMI-RIA system

As the interest for intelligent facilities in classroom environments grows, the approaches towards this issue expand, covering a wide range of topics. During the past ten years several systems have been developed targeting the environmental control of the classroom. Such systems assist the presence of students and teachers in the classroom. However, their impact on the learning and the teaching process is rather limited. A second approach to the realization of the intelligent classroom involves the classification of the content and its personalized delivery to students according to their individual needs. This approach indeed supports the students by tailoring the learning process into their needs and preferences, but the role of the teacher is not addressed.

Nevertheless, in a real classroom the teacher has a significant role that can benefit from intelligent facilities. Towards this, new approaches are targeted to assisting the teacher by observing the students' activities. However, the implemented systems either do not support real-time monitoring of the students, or are context-specific, e.g., programming courses. Furthermore, the visualization of the students and the classroom remains rather simple and lacks much information.

In this thesis, the AMI-RIA system is proposed focusing on the provision of real-time, context-generic assistance tools to the teacher. The AMI-RIA system uses a number of intelligent agents to observe unobtrusively the students' activities during the learning process and generate inferences on them. The system introduces a novel technique for the visualization of the students and their activities in the classroom. Additionally, a set of tools are designed for assisting the classroom management, such as the attendance record, the testing process, the homework assignment, etc. Finally, a powerful statistics module exploits the information from the students' educational activities in order to

generate graphs providing long-term overview on the students' progress and performance.

Summarizing, the proposed system aims to provide the following facilities in the context of the intelligent classroom:

- Real-time and context-generic observation of the students' activities
- Inferencing on the students' activities to generate further information
- A visualization tool for real-time overview of the classroom
- Progress and performance monitoring of the students
- Attendance record
- Automation of classroom's everyday tasks, e.g., exercises, tests and assignments
- Statistics and graphs on students' performance
- Automation of the lesson schedule

System implementation

3.1 Architecture

The teacher assistance tools proposed on this thesis aim to bridge the gap between the students and the teacher by providing feedback about the students' activities during the educational process. To achieve this, a modular architecture (Figure 7) is introduced that consists of two major components, the *Desk Monitor* that is an intelligent agent deployed on the students' desks to collect the data and the *Teacher Assistant,* which is deployed at the teacher's desk assisting him to keep track of the classroom. Regarding the intercommunication of these components, the system relies on a generic services interoperability platform, named FAMINE (FORTH's AMI Network Environment), which has been implemented in the context of the ICS-FORTH AmI Programme. FAMINE meets the requirements that a middleware system for Ambient Intelligence should address as reported in [23]. These requirements involve heterogeneity integration, synchronous and asynchronous communication, resilience, security and ease of use. Famine provides the necessary functionality for the intercommunication and interoperability of heterogeneous services hosted in an AmI environment. It encapsulates mechanisms for service discovery, event driven communication and remote procedure calls, supporting a plethora of programming languages and frameworks. The following sections will describe the components of the system and the interconnections between them.

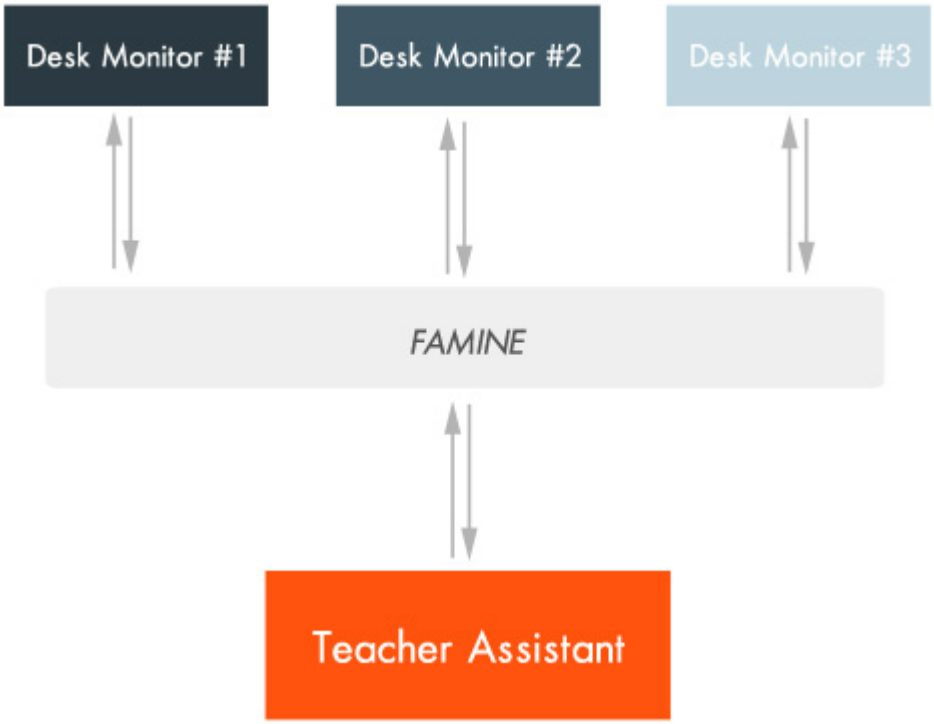


Figure 7: The AMI-RIA system architecture

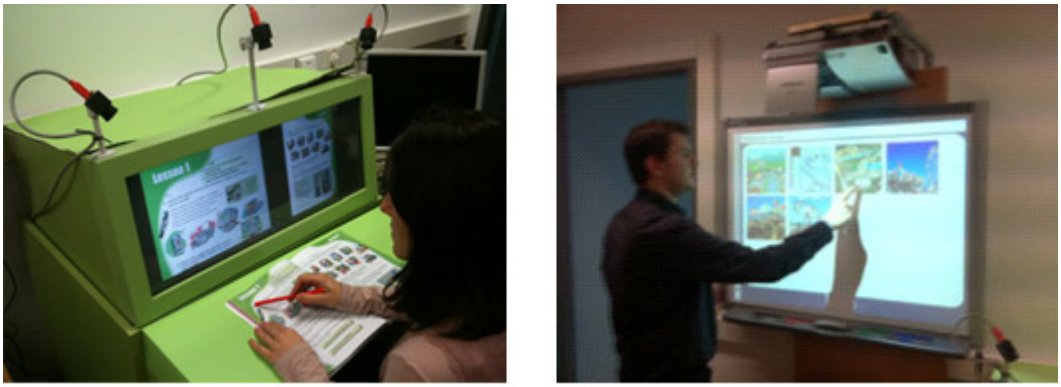
3.2 Desk Monitor architecture

The *Desk Monitor* is an agent type module that constitutes the entry point of the system. An instance of the *Desk Monitor* is deployed in every student desk aiming to observe the student's activities and form models of them. The models consist of personal data like name, grade, profile picture and also include information about the current activities of each individual. Furthermore, the *Desk Monitor* is aware of some aspects of the classroom's status, such as the current lesson being held in the classroom.

3.2.1 Data collection from the students' desks

The AMI-RIA system is targeted to an intelligent classroom environment as it is realized in the context of the ICS-FORTH AmI Programme. The implemented prototype of the intelligent classroom is based on a backbone

infrastructure called *ClassMATE* [4], which monitors the ambient environment and makes context-aware decisions in order to assist the student in conducting learning activities. *ClassMATE* is responsible for orchestrating the various artifacts that exist in the classroom, such as the augmented desk [24] and the commercial interactive whiteboard named SMARTboard presented in Figure 8.



(a) (b)

Figure 8: a) The augmented student desk b) The SMARTboard

Additionally, a learning support system is designed and deployed on the augmented desk of the smart classroom prototype discussed above. This environment, named SESIL [25], is able to perform recognition of book pages (see Figure 9) and of specific elements of interest within a page, as well as to perceive interaction with actual books and pens/pencils, without requiring any special interaction device.

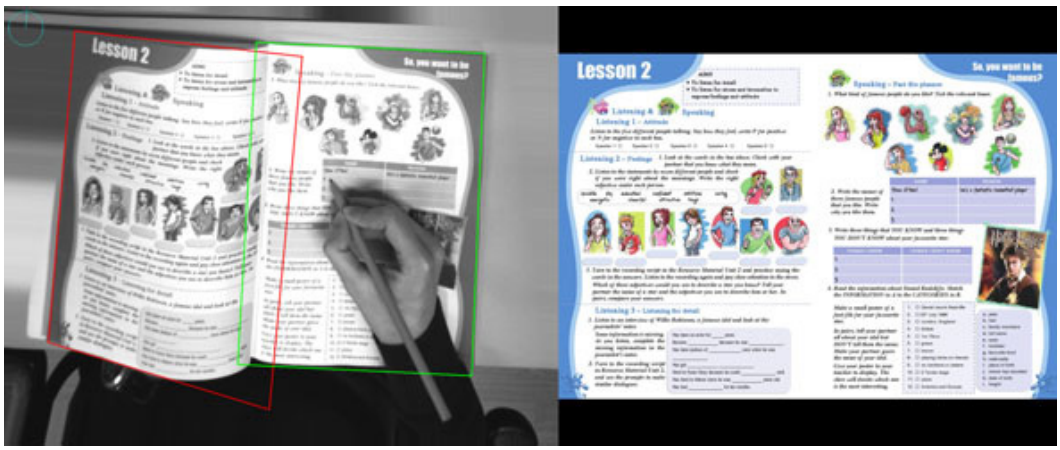


Figure 9: Recognition of book pages and their localization on the SESIL system

A functional prototype of SESIL has been deployed on the smart school desk presented in [24], which enhances a standard school desk. The enhancements constitute two additional pieces of hardware. First, it is a piece of furniture that frames a 27-inch wide touch screen and second, a triplet of calibrated cameras that overlooks the desk's workspace supporting computer vision methods, which serve the same purpose. Both enhancements are aimed to support the interaction of the user with the system. The deployment of this system on the augmented desk gives the opportunity to track the students' activities unobtrusively during the learning process. The information gathered by SESIL goes through ClassMATE and forwarded to the *Desk Monitor* module.

3.2.2 Desk Monitor components

The *Desk Monitor* consists of five components, including the core (Figure 10). *ClassMATE* forms events from the data collected from student desks and the SMARTboard and transmits them through FAMINE to the *Desk Monitor* module. The *Data Receiver* component provides the handlers for listening to these events and is responsible for forwarding the data included in the events to the core component.

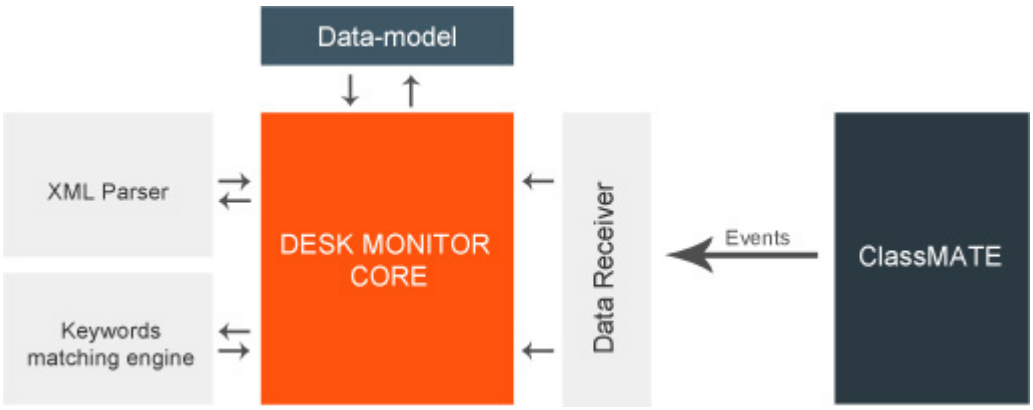


Figure 10: Desk Monitor architecture

The *Core* deploys the data received to form a model of the student based on his/her profile and activities. The student's model is stored internally in a data-model using RDF triplets. As the data are stored in the data-model, the core

component evaluates a set of predefined rules to generate inferences about the students. The rules are implemented and stored in a separate external file so as the logic is totally separated from the program. This way any non-programmers can edit the rules without the need of changing the source code of the program. Finally, the *Core* component is responsible to form events containing the data and inferences about the student's activities and transmit them to the teacher's desk. As the students may change their desks during the semester, no permanent storage exists for the RDF data-model constructed by the *Desk Monitor*. When the student leaves his/her desk, the collected data are erased.

An exercise or test in the electronic form of the book presented on the augmented desk is implemented in a defined XML schema. When a student starts a learning activity, e.g., an exercise, *ClassMATE* sends to the *Desk monitor* a reference link to the corresponding XML file, which contains among other data, the type of activity e.g. multiple-choice exercise, the questions and the answers for each question, if exist. The *XML Parser* component is responsible for parsing such files and storing this information in internal data structures for immediate access by the *Core*. Having the questions and answers extracted, the desk monitor can decide if a student is answering correctly or not in a question during an exercise or test.

When a lesson starts, the *Desk monitor* is notified about the relevant book or books and the relevant pages regarding the topic that is going to be discussed in the classroom. This information is stored in the classroom's lesson schedule and handled by ClassMATE. Apart from the book titles and pages, when a lesson session starts ClassMATE sends a list of keywords extracted from the elements existing in the corresponding relevant to the topic pages. At the time a student opens a page, the *Desk Monitor* checks if this page is relevant to the current lesson. In the case that the page the student is looking at seems to be irrelevant to the current context, a secondary process takes place to check this. The *Keywords Matching* component is used to compare the relevant pages keywords with the keywords from the student's current page. The score of this search is used to identify if that page contains keywords relevant to the current topic

discussed and thus, it will indicate if the current student's page is relevant to the lesson. The entire process of matching is discussed in section 3.5.

3.3 Teacher Assistant architecture overview

The *Teacher Assistant* is the part of the AMI-RIA system that is deployed on the teacher's computer or tablet and aims to exploit the information originating from the student desks. It consists of nine components layered in parallel as presented in Figure 11.

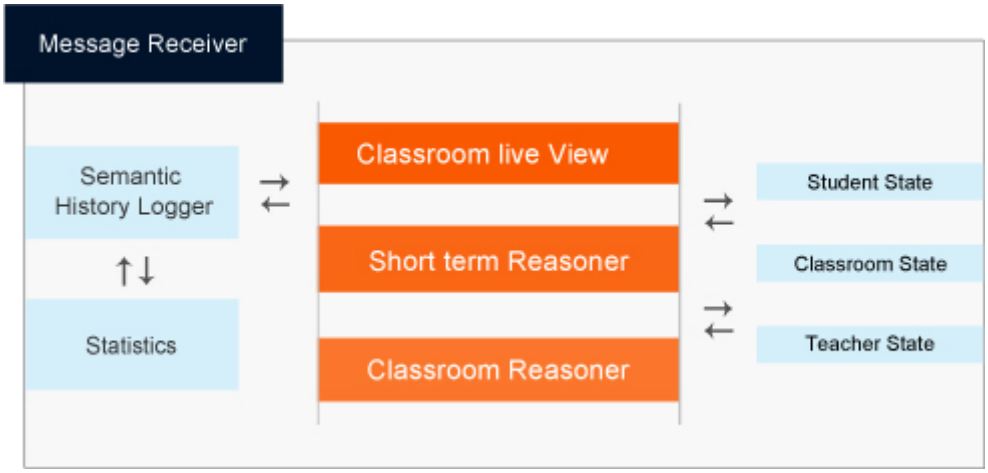


Figure 11: The Teacher Assistant architecture

The *Message Receiver* component is used to listen and handle the messages generated by the student desks and transmitted through the middleware infrastructure. It is responsible for identifying if a message is originating from a student desk and if it is, from which desk. Each message that goes through the *Message Receiver* refers to a specific desk, and therefore to a specific student. The message receiver component implements an event propagation mechanism to transmit the messages to the remaining components of the *Teacher Assistant* application.

Student, Classroom and Teacher states represent the corresponding entities in the intelligent classroom world. A *Student State* instance is generated for every registered student in the classroom and stores information regarding

the student and his/her latest activities. The *Student State* offers a quick and immediate storage space for the information needed in real-time by the system. The *Classroom State* stores information regarding the classroom status, such as the number of present students, the current course, the attendance record, etc. Like the *Student State*, it offers an immediate storage space and provides the information required at run time. The *Teacher State* is used to keep information about the current teacher in the classroom such as personal data and preferences that are used for the personalization of the monitoring process.

At the core of the *Teacher Assistant* module there are three major components, the *Classroom Live View*, the *Short-Term Reasoner* and the *Classroom Reasoner*. The *Classroom Live View* offers a real-time overview of the classroom and generates notifications and alerts based on the data and inferences generated by the students' desks. The *Short-Term Reasoner* component is responsible for observing the students' activities in short time periods. If a pattern of activities is identified then the short-term reasoner generates notifications for the teacher. The *Classroom Reasoner* is a component targeted to observe the classroom as a whole. It generates notifications and alerts based on the classroom's tendencies that the teacher should be aware, for example, when the majority of the students have problems with a specific exercise.

The main and permanent storage mechanism of the system is the *Semantic History Logger*. This component listens to all the events from the message receiver, generates RDF triplets based on the data arriving and stores them in a RDF data-model. The data are inserted on the data-model according to a defined ontology that describes entities hierarchies (taxonomies).

Finally, the *Statistics Module* exploits the data that are stored in the history log to generate graphs and statistics about the students' activities. It offers an environment to assist the teacher keep track on the students' progress and performance over short or long time periods.

3.4 Data representation and reasoning

The AMI-RIA system is based on the extraction of valuable information for the teacher regarding the students' activities during the educational process. Therefore, a technology is required that can offer rich semantic representation, entity hierarchies and mechanisms for inferencing and information retrieval. Towards this, RDF [26] and RDFS [27] were utilized for the information representation, the Euler engine [28] in combination with Notation3 [29] rules for the inference extraction and finally SPARQL [30] queries for the information retrieval. The implemented rules exist in external files separating this way the logic from the program. Due to the large amount of information ending in the semantic data-model, the available data are stored in a MySQL database implementing this way a permanent storage and at the same time keeping the performance at high levels.

3.4.1 RDF

The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata datamodel. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax formats.

RDF is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, known as triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, to represent the notion "Nick has started exercise one" in RDF, it is sufficient to declare *Nick* as the subject, the action *started exercise* as the predicate and *exercise one* as the object.

3.4.2 RDF Schema

RDF describes resources in terms of named properties and their values. The RDF Vocabulary Description language, RDF Schema (RDFS) describes vocabularies used in RDF descriptions. RDF vocabularies describe properties, classes of resources and relationships between them.

Classes in RDF Schema are much like classes in object oriented programming languages. This allows resources to be defined as instances of classes, and subclasses of classes. Classes are themselves resources and are usually identified by URIs. A resource may be stated to be a member of a class using the “rdf:type” property.

Properties in RDFS are relations between subjects and objects in RDF triples, otherwise called predicates. All properties may have a defined domain and range. The domain of a property states that any resource that has a given property is an instance of some class. The range of a property states that the values of a property are instances of a class. If multiple classes are defined as the domain and range, then the intersection of these classes is used. The taxonomy of classes is formed by the property “rdfs:subClassOf” and the taxonomy of properties is formed by property “rdfs:subPropertyOf”.

To develop the hierarchies needed for the entities, Protégé, an open source ontology editor and knowledge base framework described in [31] was used. Protégé offers a suitable environment for modeling the entities and forming SPARQL queries.

3.4.3 Semweb.NET

SemWeb.NET [32] is a Semantic Web/RDF library written in C# for Mono or Microsoft's .NET. The library is capable of reading and writing RDF in both RDF-XML and Notation3 format. SemWeb.NET provides mechanisms for keeping RDF in persistent storage (memory, MySQL, etc.), querying persistent

storage via simple graph matching and SPARQL, and making SPARQL queries to remote endpoints.

Furthermore, SemWeb.NET provides RDFS and rule-based reasoning over the data that are stored in the data-model, based on the Euler proof mechanism. Euler is an inference engine supporting logic-based proofs. It is a backward-chaining reasoner (working backward from the goals) enhanced with Euler path detection. Euler engine is used in this thesis to generate inferences based on the facts that exist in the data-model, the taxonomies that have been defined for the entities and finally, a set of rules. Figure 12 shows an example of a rule for the Euler engine written in the Notation3 format. This example rule checks if a student used more hints than the normal amount.

```
{
?student        a                                   student:Student.
?student        student:Current_Exercise            ?exercise.
?exercise       student:HasQuestion                 ?question.
?question       student:Hints_Used_For_Question     ?hints.
?hints          math:greaterThan                    "2"^^xsd:int.
}
=>
{
?question       student:Hints_Overuse               "true"^^xsd:boolean.
}.
```

Figure 12: Example of Notation3 rule

3.5 Keywords matching engine

The identification of when a student is off-task during a course is a key feature of the teacher assistant system proposed in this thesis. One method for tracking when a student is off-task is to check whether the material that is opened in his/her desk is relevant to the current topic discussed in the classroom.

For each learning session, a set of relevant material, books and pages, is predefined and checked against the material that a student has opened in his/her desk. Thus, it is an easy task to identify if a student has opened some

non-relevant material just by comparing book titles and page numbers. Although this method is safe and effective, it has the limitation that all the available material needs to be categorized accordingly at any time. This means that if a student opens a new book that is not yet classified and listed as relevant, the system, may consider that it is non-relevant to the current session and thus, identify the student to be off-task. Towards this end, a solution was investigated to enhance the process discussed above by identifying if a page in a book is relevant to the current session based on its content.

Each page on the digital version of a book constitutes of a set of elements including text, images and activities. These elements are described by a number of metadata keywords that are used by ClassMATE in order to provide personalized content to the students. The union of the metadata and text included within a page provides a description of the content appearing on it. Therefore, they can be exploited to offer a rough estimation of whether a student's current opened page provide content relevant with the topic discussed in the class. The procedure of the keyword matching consists of two phases, one for indexing the keywords and text from the listed relevant pages and one for searching the current's page keywords against the indexed ones.

The indexing phase takes place when a learning session starts. The keywords and text from the listed relevant pages are extracted by ClassMATE and transmitted to the *Desk Monitor* module. The implemented *Keywords Matching* component receives them and starts the process of removing the stop words. The stop words are extreme common words, which are filtered out prior to processing of natural language. The list of stop words used in the matching engine described here, includes some of the most common short function words[4], such as *the*, *is*, *at*, *which*, *on*, etc. and some common lexical words[5] such as *because, both, come, do, particular,* etc. The elimination of the stop words is used both for performance optimization and for increasing the effectiveness of

[4] http://en.wikipedia.org/wiki/Function_word

[5] http://en.wikipedia.org/wiki/Lexical_word

the searching process. The list of stop words used in this component includes a total number of 659 words.

Additionally, a further optimization is incorporated in the indexing procedure namely, stemming. Stemming is the process of reducing inflected or derived words to their stem, base or root form. The stem need not to be identical to the morphological root of the word and usually it is sufficient that related words map to the same stem even if this stem is not itself a valid root. A stemmer for English for example, should reduce the words “fishing”, “fished”, “fish” and “fisher” to the root word “fish”. This, results to increase the overall search performance by reducing the number of words in the index and achieve better accuracy on the searching.

There are several types of stemming algorithms which differ in respect to performance and accuracy. In this thesis, an implementation of the Porter Stemming Algorithm [33] is used. The Porter stemmer is a conflation stemmer developed by Martin Porter at the University of Cambridge in 1980. It is a linear step stemmer consisting of five steps. Within each step a set of rules are checked against the word and once a rule passes its conditions the suffix is removed and the control moves to the next step. The porter stemmer is a very widely used algorithm by many applications and probably the stemmer most widely used in information retrieval (IR). Implementations of the Porter stemmer are available in Java, C, C# and many more.

The keyword processing algorithm has a computational complexity of O(N). Table 1 presents the processing time needed and the results of the index process running on a modest computer.

Words number	Words number after processing	Processing time
100	40	18ms
200	79	19ms
500	160	25ms

1000	255	33ms

Table 1: Processing time for the keywords indexing procedure

Once the indexing phase is completed, text queries can be performed against the index. When a student opens a book page the keywords are extracted and transmitted to the *Desk Monitor.* The procedures of removing the stop words and stemming are performed as well to this set of keywords that is considered to be the query. The custom searching process implemented, includes checking all the words from the query against the indexed words. When a keyword appears both on the query words and the indexed words then a hits counter increases by one. By the end of the checking procedure the percentage of the query words appearing in the index is calculated using the following formula:

$$Score = \frac{Number\ of\ query\ words\ appearing\ in\ index}{Total\ number\ of\ query\ words}$$

This score indicates what fraction of the query appears in the index and it is used as a measure to indicate if the current page is relevant with the topic discussed in the class. This assumption is based on the fact that chapters covering the same topic will use a set of identical words to describe it. Thus, a threshold is calculated through empirical evaluation, which is used to define if the score indicates that a page is relevant to the current session. This threshold is set at this point to 0.35 and is editable through the rules external file. Further evaluation of the system is required under real-world situations in order to precisely define the threshold for this score.

Student monitoring

The teacher assistance tools proposed in this thesis are based on the observation of the students. The monitoring involves collecting data about the students' activities on the augmented desk and the SMARTboard during the learning process, e.g., start an exercise, answer a question, browse a multimedia gallery, etc. These data constitute the base of a reasoning process that will generate valuable insights regarding the students. The collected data along with the new information generated through the reasoning process are transmitted to the teacher's desk in order to assist him/her to keep track of the classroom and identify problems that need to be addressed.

Recent studies [34], [35], [36] revealed the potential of expanding the monitoring process on the students' physical activities by observing their movements, body posture and facial expressions. These characteristics were used to generate a model of the student's emotional status in order to help the system understand if he/she is feeling boring, frustrated, excited etc. However, privacy concerns arise from this type of monitoring because of the large number of cameras and sensors required to observe the students. Furthermore, the research conducted in this field revealed that there is no precise algorithm, which would decide when a student is bored, frustrated or excited based on the posture or movement. Having a weak algorithm will lead to having many false positives. Therefore, it would be a hard task for the teacher to keep track of all the students when he gets a fair percentage of false information. Due to its limitations, this type of observation called affective monitoring is not further investigated in the context of this thesis.

The following sections will present the activities to be monitored and the rules that generate the inferences that are used to keep the teacher informed about the classroom state.

4.1 Identifying the activities

The *ClassMATE* infrastructure for the smart classroom in collaboration with the *PUPIL* system offers an interactive environment for the students. Every time the student performs an activity on the augmented desk, interacts with the desk's interface or the SMARTboard, the actions are captured and transmitted to the AMI-RIA system through the FAMINE middleware.

Captured activities form a model that represents the current status of a student. This model is stored in an internal data-model using RDF triples. The collected data are stored in a semantic way, so as to easily apply reasoning to them with the use of the Euler proofing mechanism. No persistent storage is used at this point as the size of the model remains always limited. This is because new statements from the received data are not appended in the data-model but in fact, they replace existing ones. The memory storage achieves better performance than the persistent one for reading, writing and querying the data. Semweb.NET provides extended benchmarks[6] regarding the performance of the library in respect of the various methods of storage (memory, MySQL, SQLite, etc).

The data from the students' activities are presented in the following sections categorized according to the type of activity:

1) **Logging:** Students log in their desks by using their personal notepad on which an RFID tag is attached. The desks scan the tag to identify the students.

2) **Studying:** This category includes actions relevant to the studying process. Activities to be monitored are:

   - Open a book
   - Go to a page

[6] http://razor.occams.info/code/semweb/semweb-current/doc/bsbm.html

3) **Exercises:** Activities regarding the exercises on a book are included in this category. Such activities are:

   - Start an exercise
   - Answer a question
   - Request help (hints) for a question
   - Submit an exercise
   - Skip an exercise

Additionally, when an exercise starts, ClassMATE sends some additional information about the exercise describing aspects of the learning object, including:

   - The exercise type (Multiple-choice, fill-in the gap, etc.)
   - The exercise level (Easy, Medium, Hard) as it is defined by the "Difficulty" attribute in the LOM specification [37]
   - The exercise topic as it is defined by the "Keyword" attribute in the LOM specification
   - The educational learning time also defined in the LOM specification describing the approximate or typical learning time it takes to work with or through for the typical intended target audience
   - The learning object URI referring to the actual learning object stored in the classroom's repository

4) **Tests:** The testing process is designed as an extension to the exercise. It includes activities that are initiated either by a student or by the teacher. This process includes the activities:

   - Start a test (teacher)
   - Answer a question (student)
   - Submit a test (student)
   - Force the submission (teacher)

5) **Searching for relevant info (Multimedia gallery):** When the student finds an interesting image in the book, he/she can click on it and view more

images, video and relevant information to further explore the topic. Actions included in this process are:

- Open a multimedia gallery
- Navigate through multimedia
- Share multimedia on board
- Close the gallery

4.2 Student activities reasoning

A key feature of the *Desk Monitor* is the ability to generate inferences about the students' activities. The logic for the inferencing process is defined by a set of rules written in the Notation3 format (Figure 13). The Euler engine exploits the rules and the available data to generate the inferences that will provide to teacher valuable insights about the students. Furthermore, a custom ontology named *Classroom* is defined to describe the taxonomies over the available data and enhance the reasoning process. The *Classroom* ontology is described in section 5.2.1.

```
#********** IN CONTEXT RULE 1 *********************
{
?student        a                                   student:Student.
?student        student:On_Relevant_Book            "true"^^xsd:boolean.
?student        student:On_Page_Range               "true"^^xsd:boolean.
?student        student:On_Exam                     "false"^^xsd:boolean.
}
=>
{
?student        student:In_Context                  "true"^^xsd:boolean.
}.
```

Figure 13: The "In context" rule

The following sections are describing the rules used for the inferencing process. These rules are created to reveal the potential of the concept and can be customized at any time even from non-programmers. The majority of the defined rules are based on studies relevant to student behavior monitoring such as [38], [39], [40] and [41] . The rest of them are custom rules created in the

context of this thesis. An extended evaluation is required to take place as a future step in order to estimate the rules' accuracy and the efficiency of the system.

4.2.1 Identify an off-task student

According to Carroll's Time-On-Task hypothesis [39], the longer a student spends engaging with the learning material, the more opportunities the student has to learn. Therefore, if students spend a greater fraction of their time engaged in behaviors where learning from the material is not the primary goal, they will spend less time on-task and as a result learn less. Baker et al [38] argue that off-task behavior has an impact on the students' performance and investigate the impact on learning caused by different off-task behaviors. The results of this study indicate that the frequency of off-task behavior is a good predictor of the students' performance; however, different types of off-task behavior result different negative correlation to learning. Additionally, Cocea et al in [40] resulted that off task behavior appears to be associated with poorer learning at an aggregate level.

An off-task behavior in the classroom can be defined in several ways. Having in mind that the AMI-RIA system monitors the activities of a student on the augmented desk, an off-task student can be defined based on the material on his/her desk. Thus, a student having on the desk an unrelated book, page or exercise is considered to be off-task. The term out-of-context is used to describe this specific type of off-task behavior.

A student is considered to be out of context when the book and the page he/she is currently on are not listed as relevant with the topic currently discussed in the classroom. If the current opened book is not listed as relevant to the topic, the keywords from that page are extracted and matched against the keywords from the listed relevant pages. If the matching score is bellow a specified threshold the current material is identified as irrelevant.

4.2.2 Identify an inactive student

There are times when a student starts working on an exercise but at some point he/she gets bored or distracted and gives up the effort. Inactivity is defined as a type of off-task activity where the student does not interact with the learning object when he/she is supposed to. Inactivity indicates that a student is disengaged with the task and this is a reasonably good predictor of performance [38]. If the teacher is aware of this situation, he can focus on this student and try to identify the problem. Additionally, if the teacher realizes that a specific exercise is boring or too hard to solve, he can replace it with another one, adapting this way the teaching process.

Towards this end, a mechanism is proposed for identifying the students who are getting inactive during an exercise. When a student starts an exercise, this mechanism estimates a maximum time interval between two actions. Every time the student interacts with the exercise, e.g., answers a question or uses a hint, the timer resets and starts all over. If the timer expires, the student is considered to be inactive and the teacher is notified accordingly.

The maximum interval used is based on the typical learning time for this learning object. The typical learning time is part of the standard for the learning object metadata and is targeted to describe the approximate or typical time it takes to work with or through this learning object for the typical intended target audience. This time is provided in the ISO duration format (PT1H30M), either by the learning object's author or by the teacher that use the object in the classroom. Ideally, the questions of an exercise should be defined as individual learning objects too. However, it is common to define as learning objects the whole exercises/tasks. Therefore, to estimate the typical learning time for a single question the typical learning time of the exercise is divided equally to the number of questions assuming that the difficulty is roughly the same between the questions.

However, not all the students interact with the exercise at the same pace; each one has his/her individual needs. In order to estimate a more personalized interaction time the learning level of each student is used. To achieve this, Algorithm 1 is implemented in the form of Notation3 rules and targets to estimate the maximum interaction time based on the following data:

1. Educational typical learning time of the exercise(Approximate or typical time it takes to work with or through this learning object for the typical intended target audience)
2. The student's average score on the exercises of this course

Algorithm 1 works as follows: Firstly, the typical learning time for the exercise is divided equally to the number of questions. Secondly, the time for a single question is adjusted according to the average score of this student in this course's exercises. The lower the average score is, the greater the additional time should be in order to give the student the time to evaluate his/her answer and respond on time. In the case a student has an average score of 100% there will be no additional time, however in the case a student has an average score of 0% the additional time will double the total time for a single question.

```
Input: Typical learning time tlt for the exercise, Number of questions n,
Average score on this course averageScore
Output: Maximum interaction time interaction
  1: timeForQuestion = tlt / n
  2: timeAverage = ((100 – averageScore) / 100) * timeForQuestion
  3: interaction = timeForQuestion + timeAverage
  4: return interaction
```

Algorithm 1: Find the maximum interaction time

4.2.3 Identify a student that requires attention during an exercise

One of the most interesting situations in the classroom is the exercising process. When the students are working on an exercise, it is an important task to identify any problems. The teacher should be able to detect in real-time the learning gaps of each student and intervene when needed to provide help. A set of rules is defined that aim to estimate if a student is facing problems with an exercise.

Student disengagement

According to Beck [41], time on-task is an important predictor of how well students learn a skill. Therefore, it is important to ensure students are engaged in the learning process. If students are disinterested, it does not matter how pedagogically appropriate the material is, learning will not be sufficient. Frustration and boredom are some of the reasons for the student's disengagement during learning. Two set of rules are defined within this context in order to identify situations where a student is frustrated performing poorly or is bored and starts gaming the system.

A student is identified to be frustrated when he/she use the available help but not take advantage of it. This means that a student tries to answer a specific question and uses all three available hints but he/she does not achieve to answer correctly.

Some students interact with exercises according to a set of non-learning-oriented strategies. This set of strategies described in [38] as "gaming the system" involve behavior aimed at systematically misuse the system's help in order to advance in exercise instead of actively thinking about the material. A set of rules targeted to "gaming the system" behavior have been created to check if the students quickly and repeatedly asking for help until they get the maximum one, which will help them advance. The system check the timestamps of the hint use actions and if the interval between two hint use actions in a question is too short the student is considered to "gaming the system".

Problem on exercise completion

Detecting a student who faces problems with an exercise could be a difficult task. A way for identifying student's difficulties in an exercise is to grade his/her effort and generate a pass/fail indicator based on the grading scale[7] widely used in European countries. The maximum fail grade at the most cases is set to 49% of the total score (Algorithm 2).

```
Input: Student's score on this exercise score, Teacher preference on
threshold threshold
Output: True if it is estimated that the student faced problem, false if he
did not
 1: if score > threshold
 2:     return false
 3: else
 4:     return true
```

Algorithm 2: Identify a student having problem with an exercise based on score

Identifying a student who is in difficulty based just on the current score is safe but not always sufficient. If a student scores 60% - 70% while his/her average score is 90% - 100%, it seems that he/she is facing problems with this specific exercise. The minimum deviation between the current score and the average score is set to 20% as the student gets down two grading levels e.g. from A to C (Algorithm 3).

[7] http://en.wikipedia.org/wiki/Grade_(education)#Europe

```
Input: Total score on the current exercise score, Default minimum deviation
problemThreshold, Student's average score on this course average
Output: True if it is estimated that the student face problems, false if not
  1: deviation = average – score
  2: if deviation > problemThreshold
  3:    return true
  4: else
  5:    return false
```

Algorithm 3: Identify a student having problem with an exercise based on the relative performance

Problem on exercise skipping

There are times when a student starts an exercise but at some point he/she decides to skip it. The student could just simply do not want to do the exercise or might be not confident enough to go through it. In order to identify such situations the system exploits the information gathered during the student's attempt to complete the exercise. In particular, two rules are created based on the amount of help used.

A student who skips an exercise is considered to face problems if he/she tried to complete it but at the end did not make it. An indicator for this situation is the use of hints. If the student used at least one hint for every unanswered question it means that he/she made an effort on completing the exercise. On the contrary, if a student skips an exercise while there are unanswered questions on which the student did not use at least one hint indicates that he/she did not make a strong effort to complete the task as there is some extra help that he/she did not take advantage of.

4.2.4 Extendibility

The Notation3 format, which is used for creating the rules, has been designed to be human friendly. Thus, even non-programmers can easily create

new rules generating this way, new inferences. New data that may be available in future releases of the student's augmented desk can easily inserted in the data-model through a trivial process and therefore new rules could be created to generate further inferences.

Teacher Assistant

The information extracted during the educational process about the students' activities is transmitted through FAMINE middleware to the teacher's desk. There, a tool named *Teacher Assistant* is developed and deployed in order to assist the teacher in keeping control of the classroom and therefore giving him/her the potential for intervention either by providing guidance to the students or by adapting the teaching process to the corresponding audience. The *Teacher Assistant* is a powerful application presented to the teacher through a friendly and easy to use interface. The following sections will describe the flow of the data originating from the student desks, the management and storage of the information collected and the features supported by the *Teacher Assistant.*

5.1 Data flow

Every student desk in the smart classroom has a unique identification string and is capable of transmitting messages to several artifacts, such as the teacher's desk. The messages originating from a student desk are marked with this unique ID allowing the system at the teacher's side to identify the sender of each message.

When a student logs in a student desk, the system pairs this student with this particular desk for the current session. Therefore, every active student desk in the classroom is attached to a specific student. A hash map is implemented into the message receiver component of the *Teacher Assistant*, which matches each desk ID with the corresponding student. When a student desk sends a "log in" message, its id registers into the map and a new instance of the *Student State* class is instantiated to represent this student.

A message arrived at the message receiver module and identified successfully is forwarded to the remaining components through an internal

event propagation mechanism. This way, information is delivered asynchronously everywhere. A component of the *Teacher Assistant* can register to some or all the events and listen to the corresponding messages. For example, a component registers for the "book-changed" event and starts listening to the messages arriving when a student changes a book.

5.2 Data management

In the intelligent classroom environment within the context of this thesis, three major entities can be identified, the student, the teacher and the classroom. Three classes are defined to represent the corresponding entities namely, the *Student State*, the *Teacher State* and the *Classroom State*.

For each student in the classroom an instance of the *Student State* class is instantiated to store profile information and the latest activities performed. For example, in a *Student State* instance one can find the name of the corresponding student, his average score in this lesson, the exercise his is currently working, etc. Everything that is displayed in real-time to the teacher must exist in this instance. In order to achieve the real-time nature of the system, all the data to be presented to the user-interface are available on the dynamic memory.

For the teacher, a single instance of the *Teacher State* class is used to represent the current teacher in the classroom. The *Teacher State* class is implemented following the Singleton design pattern. The Singleton pattern [42] ensures that a class has only one instance and that the instance is easily accessible. The system does not monitor the teacher's activities but instead loads and keeps his profile info and preferences regarding the student monitoring process. An instance of the *Classroom State* class is created for the classroom entity also following the Singleton design pattern. This instance stores information regarding the classroom as an educational space, such as the attendance record, the lessons schedule, the current course, etc.

Storing data on the dynamic memory helps to keep the performance of the system at high levels; however, the dynamic storage is limited and it should be carefully handled. Furthermore, if all collected data are stored on the dynamic memory they would be lost when the session ends and the application shuts down. To address this issue, a component named *History Logger* was implemented to listen to all events arriving on the teacher's desk and store the containing data in a RDF data-model, existing in a MySQL database. Over the time a rich history log is built that contains valuable information regarding the students' progress.

5.2.1 Taxonomies

Taxonomies are a fundamental part of the Semantic web notion. They are machine-readable hierarchies that enable intelligent agents to make logical inferences, thereby making information retrieval an entirely new, more sophisticated experience. The introduction of taxonomies, combined with the SPARQL query language and the Euler proofing mechanism, result in a powerful reasoning service on the classroom activities.

The RDFS classes, hierarchy and properties were created using protégé, an open source ontology editor that offers a suitable environment for modeling the entities. The implemented *Classroom* ontology defines entities appearing in the classroom such as courses, books, students, teachers, etc. and activities that represent the actions taking place.

For modeling the activities that take place in the classroom two classes named *Student_Act* and *ClassRoom_Act* are implemented. Every activity in the classroom is a subclass of one of these classes. For example, the "course change" action is subclass of the *ClassRoom_Act* class and an "exercise start" activity is subclass of the *Student_Act* class. Additionally, further categorization is applied to simple acts. An example is presented in Figure 14, where the "skip exercise" act is a subclass of the "complete exercise" act, which is a subclass of the "exercise act" which is a subclass of "student act". This way, in order to extract

all the students that submitted or skipped an exercise it is sufficient to just search for the "complete exercise" activity and the inference mechanism will check on the taxonomy and retrieve instances from both activities. In Figure 15, the class tree of the *Classroom* ontology is presented as it is visualized by the protégé module, named Jambalaya [43].

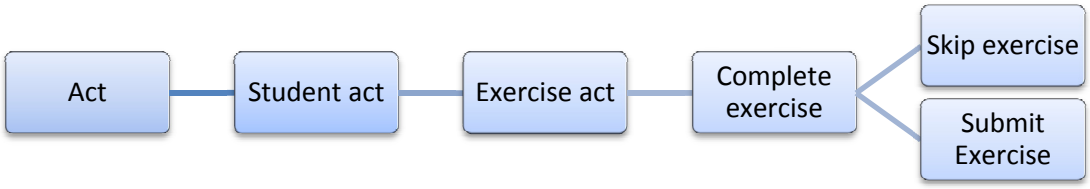


Figure 14: Taxonomy example

As semantic web notion prompts for more structured and connected data, two external popular vocabularies were used in the *Classroom* ontology that is described namely, the FOAF [44] ontology and the Dublin Core metadata [45]. Using semantic web vocabularies permits intelligent agents to make sense of the entities appearing in ontologies and the connections between them.

FOAF (acronym of Friend Of A Friend) is a machine-readable ontology describing persons, their activities and their relations to other people and objects. In the *Classroom* ontology, the student and the teacher entities are implemented as subclasses of the class Person, which is defined by FOAF. Additionally, some properties of the Person class were used to describe the students and teachers, namely, the name, the nick and the depiction. The name represents the full name of the student or teacher, the nick is a unique nickname for each entity and the depictions offers a link to an image of the corresponding person.

The Dublin Core metadata terms are a set of vocabulary terms, which can be used to describe resources for the purposes of discovery. The terms can be used to describe a full range of web resources: video, images, web pages etc. and physical resources such as books and objects like artworks. In the *Classroom*

ontology, some of the elements that are included in the Dublin Core metadata element set are used to describe the books used in the classroom. In particular, the metadata elements used are the “Title” and “Subject”.

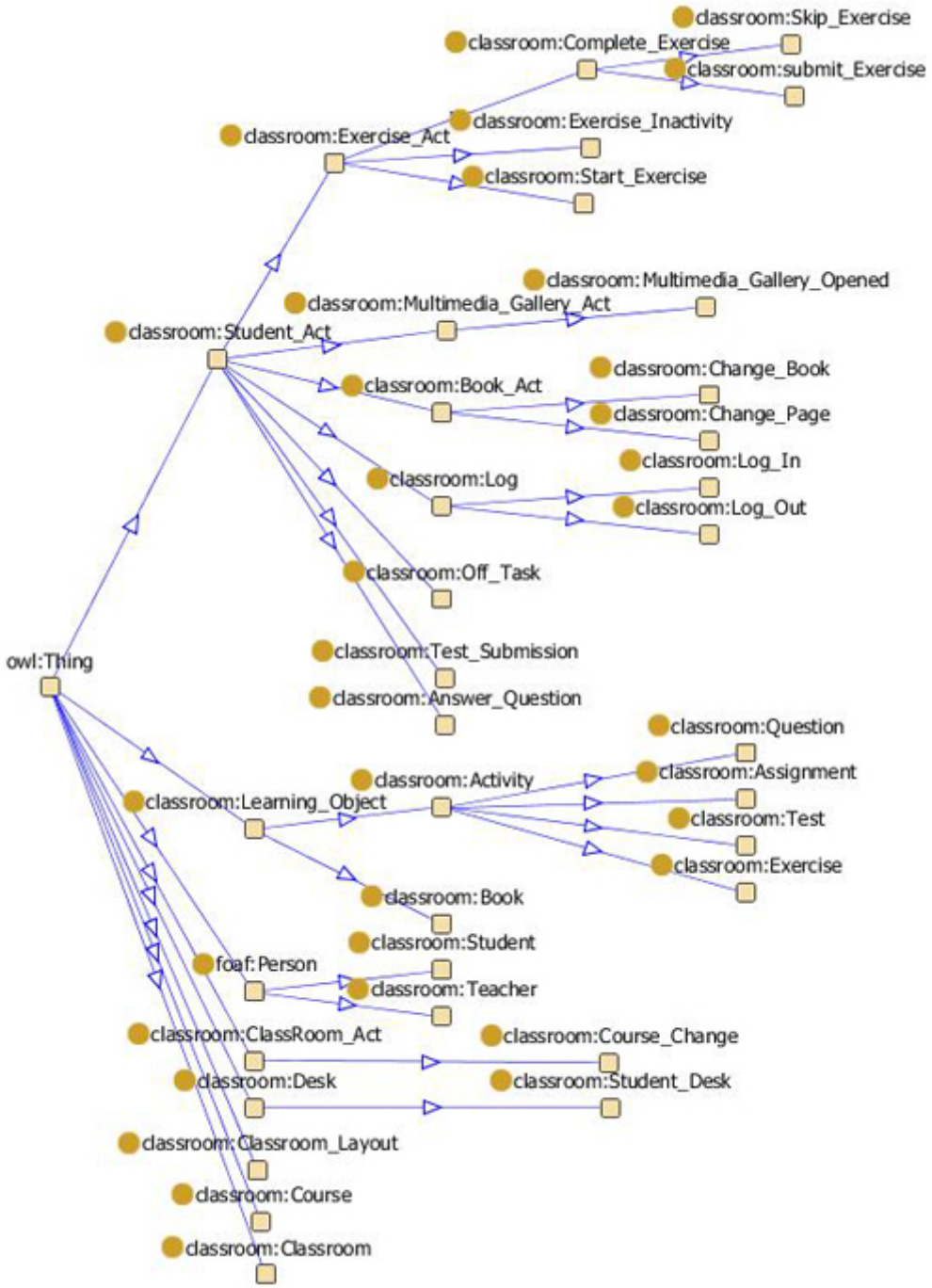


Figure 15: A full representation of the classroom ontology

5.3 The design

The *Teacher Assistant* is designed to be easy to use and look friendly to the teacher. The background displays a wooden canvas, which contains a number of cards that represent the desks and therefore the students. A card "becomes alive" when a student logs in the corresponding desk. From that moment, the card starts to display the student's activities. The design of the user-interface includes four major parts, the main menu at the top, the sidebar at the left containing context-aware preferences, the footer bar at the bottom and the main panel that displays the selected view.

The teacher application should be capable of displaying without any problems on personal computers and mobile devices. Therefore, it must be designed to work on several screen resolutions. Towards this end, a flexible layout was implemented that adapts to any screen resolution. Figure 17 presents the layout as displayed in a typical PC screen (1280 X 1024) and Figure 16 presents the layout on a tablet device (1024 X 768). The margins between the cards increase and decrease according to the number of desks and the maximum available resolution of the device. However, the number of desks that the interface supports is proportional to the device's maximum resolution. Table 2 presents a number of indicative resolutions and the maximum number of desks supported.

Resolution	Number of maximum supported desks
1024 X 768 (XGA)	25
1280 X 1024 (SXGA)	49
1400 X 1050 (SXGA+)	56
1600 X 1200 (UXGA)	72

Table 2: Number of supported desks proportionally to screen resolution

Figure 17: User-interface in 1280 X 1024 screen resolution (scaled)

Figure 16: User interface on a 1024 X 768 tablet device (scaled)

Two visual components are always visible to the user, the main menu and the footer. The main menu component provides the way of accessing all the features of the system. The system features are summarized in seven sections and therefore a corresponding number of buttons are implemented. From the left to the right there exist the buttons for the classroom live view, tests and assignments, statistics, lessons schedule, settings, lock application and exit application. The buttons are ordered in regards of the frequency of use. Figure 18 presents the menu visual component with the corresponding descriptions. The menu items have a descriptive tooltip about their functionality.

The footer bar displays information that is available to the teacher no matter what panel is displayed in the main part of the application, namely, the number of students in the classroom and the number of students who are out of context. Figure 19 presents the footer.

Figure 18: Application's main menu with descriptive labels (scaled)

Figure 19: The footer bar of the teacher application (scaled)

5.4 Teacher authorization

The teacher application displays sensitive information about the students' activities. Thereby, only registered teachers must have access to the system.

Additionally, each teacher has a set of preferences that should be loaded when his lessons start. Thus, a login mechanism is needed to authorize the teachers and load their preferences.

An authorization mechanism was developed using a magnetic card reader by Magtek[8] presented in Figure 20. The administration of the school is responsible to register a new teacher and provide a unique id along with a magnetic card containing this id. The registration data are stored in the semantic data-model of the classroom and can be accessed by the *Teacher Assistant* application. A teacher who enters the classroom swipes the magnetic card against the reader; the system reads the id, logs the teacher and loads his preferences. Figure 21 presents the front side of the magnetic card.

Figure 20: MagTek MagneSafe Mini card reader

Figure 21: The front side of the magnetic card for the teacher

[8] http://www.magtek.com/

When the *Teacher Assistant* application starts, a login screen (Figure 22) prompts the user to swipe his identification card. The same screen appears when the application is locked.

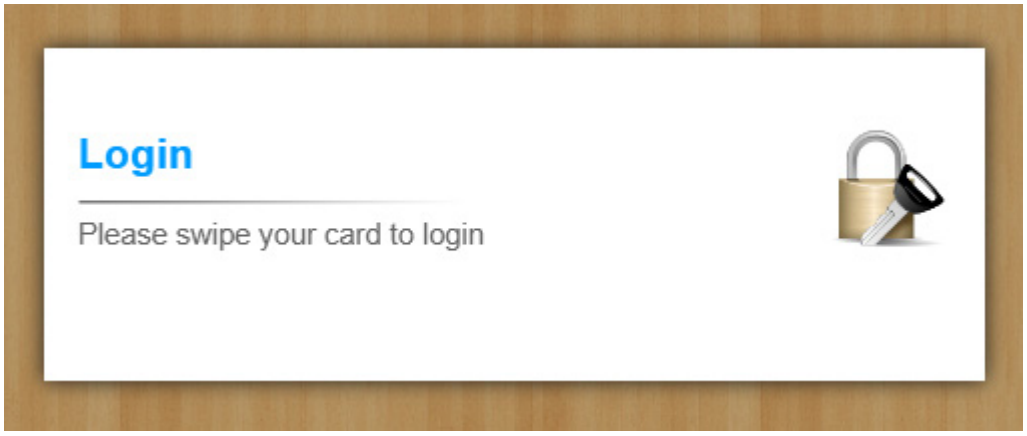


Figure 22: The login screen of the Teacher Assistant

5.5 The Classroom Live View

The *Classroom Live View* is one of the most important and extensive components of the system. It consists of a user-interface that presents to the teacher an overview of the classroom and the backend that is responsible to handle the real-time information arriving from the student desks. The *Classroom Live View* contains a number of cards that represent the student desks. When a student logs in a desk, the corresponding card "gets alive" and starts displaying relevant information.

The user-interface is structured in a way that the teacher can reach any information just by one click. The student card displays constantly the most important information and creates some secondary panels for any auxiliary information. This way, the card's size is kept small, but at the same time, all available data are reached easily.

5.5.1 Classroom layout

The classroom layout was designed so as to allow the teacher to identify a student and his/her activities at a glance. However, in a class with twenty or thirty desks this becomes a difficult task. In order to make the process of student identification easier, a digital classroom layout is defined.

The digital classroom layout is expressed in the semantic data-model in terms of desk entities with row and column properties. Therefore, when the teacher opens the *Teacher Assistant* application, a precise representation of the classroom is available on his device. Furthermore, a drag & drop mechanism is incorporated to the user-interface, which makes the teacher capable of moving around the student cards to achieve even more precise positioning. Functionality for resetting the positioning of the cards is also incorporated through keyboard shortcuts.

5.5.2 The student card

The student card constitutes a representation of a student. It contains information about his/her current activities and the information generated by the inference engine. Student cards are structured in such a way that the teacher can identify at glance the status of the student and the type of task he/she is executing.

Student states

Three possible states are defined for the students: active (in-context), inactive and out-of-context. These states are generated during the educational activities by the *Desk Monitor* module that runs on the student desks. The rules that operate this categorization are described in section 4.2. Each state is represented by one color, which is used as the border on the student card. The active state is presented in Figure 24, the inactive state in Figure 23 and the out-of-context state in Figure 25.

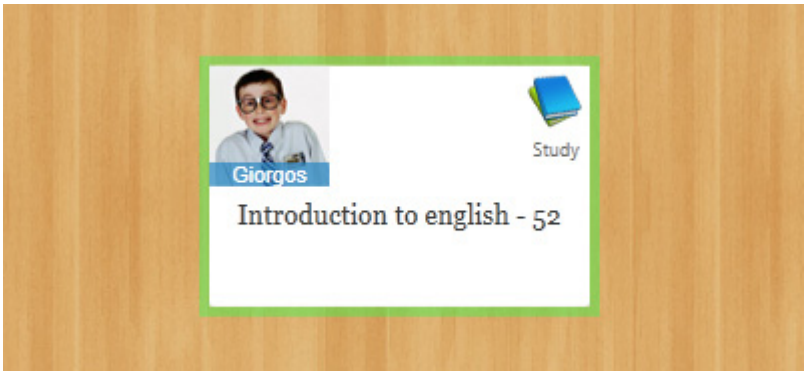


Figure 24: Student in active state

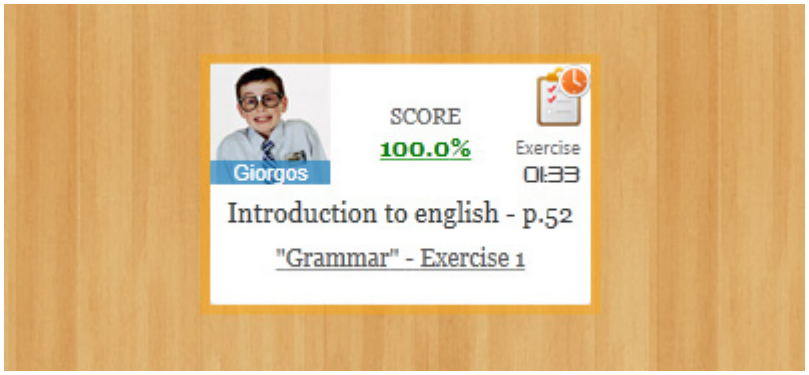


Figure 23: Student in inactive state

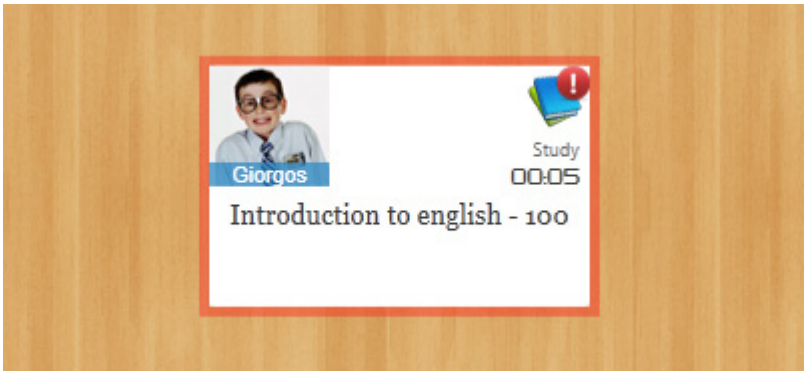


Figure 25: Student in out-of-context state

Tasks

During the data collection phase, three types of tasks are identified, namely, studying, exercising and multimedia gallery browsing. It would be very useful for the teacher to know in real-time the type of task the student is currently executing. Therefore, a relevant image is assigned to each task in order to be easy to distinguish by the teacher. Figure 26 presents the studying task, Figure 27 presents the exercising task and Figure 28 presents the multimedia gallery browsing task. The test task also exists in the classroom context but it was chosen to be represented with the same icon as the exercising task. An additional icon, which will be described later, is used to state that the classroom is on testing mode.

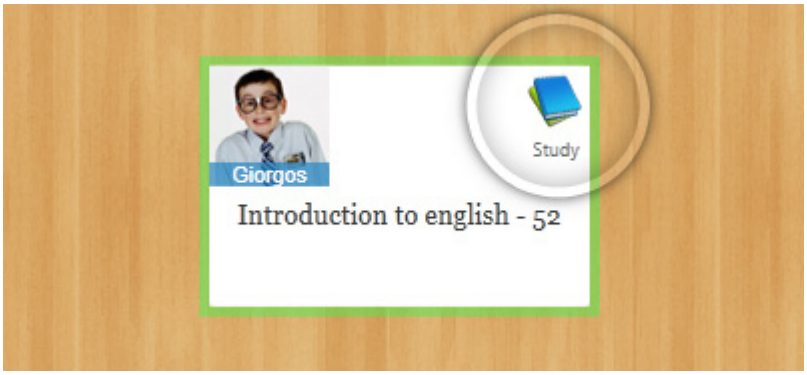


Figure 26: The study task

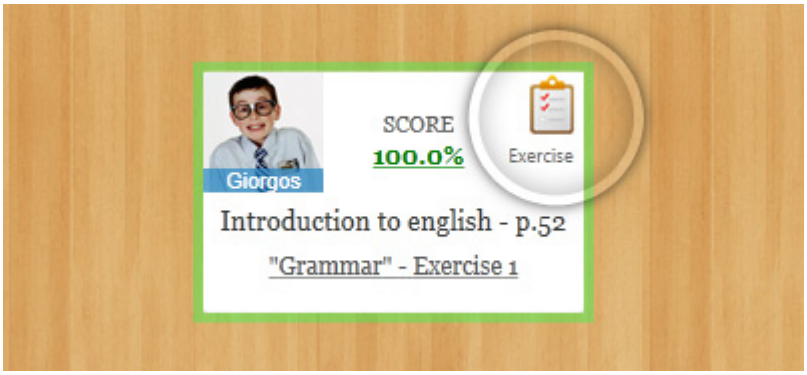


Figure 27: The exercise task

Figure 28: The image gallery task

Looking at a student card one can identify information regarding the student and his/her activities. Part of this information is always visible to the teacher, while the rest are dependents on the type of task the student is executing. The information that is always visible includes the name, the profile picture, the type of task, the current book and the current page. The remaining part includes information about exercises, image galleries and notifications.

The Action Logger

The *Action Logger* is a pop-up window that is accessible by the teacher by clicking on the students profile image. This window contains a list with the student's last activities in reverse chronological order. The teacher could catch up on a student if something else got his attention in the meantime. For example, if the teacher does not remember if a particular student completed an exercise earlier during the lesson, he/she could easily navigate in the latest activities and find the answer. Each activity is accompanied by an icon that indicates the type of activity and a timestamp.

The A*ction Logger* presented in Figure 29, serves the information in real-time. When a student performs an activity while the *Action Logger* window is opened the teacher can watch the list of activities update automatically to contain the new activity. For this to happen a feature of WPF called data binding was exploited. The list of visual components binds with a C#

ObservableCollection and updates itself when some item is added or removed from the collection.

Figure 29: Action logger pop-up window (scaled)

Exercise related information

At the time a student is working on an exercise, the layout of the corresponding student card is adapted to contain information regarding the exercise. The additional information includes the topic of the exercise, its identification number (id) and the current score of the student (Figure 27). Furthermore, two pop-up windows are introduced that contain relevant data, namely, the *Exercise Progress* window and the *Exercise Details* window.

The *Exercise Progress* window contains the questions of the exercise and the student's answers. The *Desk Monitor* module at the student desks involves a mechanism that checks the student's answers to find if a question is answered correctly or not. As presented in Figure 30, the teacher can view in real-time the questions, the student's answers marked in bold, an icon that indicates if the questions were answered correctly or not, the number of hints used and if the

system identified any problems in a specific question (frustration or gaming). Unanswered questions are marked as "Not answered yet" indicating this way the progress of the student. The teacher has access to the *Exercise Progress* pop-up by clicking the score on a student card.

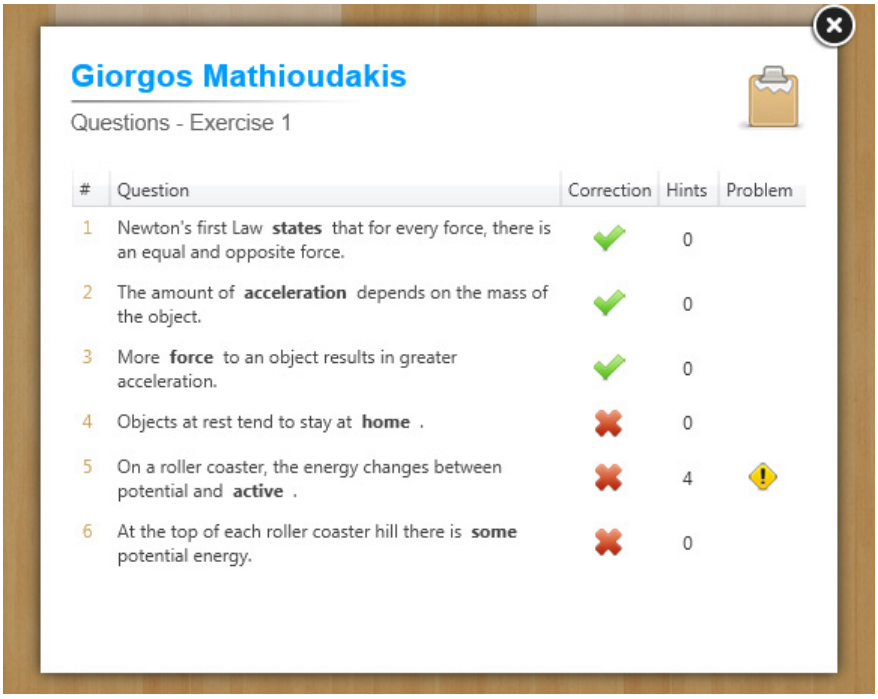


Figure 30: Exercise progress pop-up window (scaled)

The *Exercise Details* pop-up window contains information regarding the exercise, but also statistics regarding the particular student. Specifically, this window includes the type (Multiple choice, fill in the gap, etc.), the difficulty level (easy, medium and hard) and the typical learning time of the activity. In addition, some statistics for the student are presented regarding the exercise, such as the average score on this course and if the current topic is disliked or favorited. The teacher has access to the *Exercise Details* window by clicking on the exercise topic. Figure 31 presents the *exercise details* pop-up window.

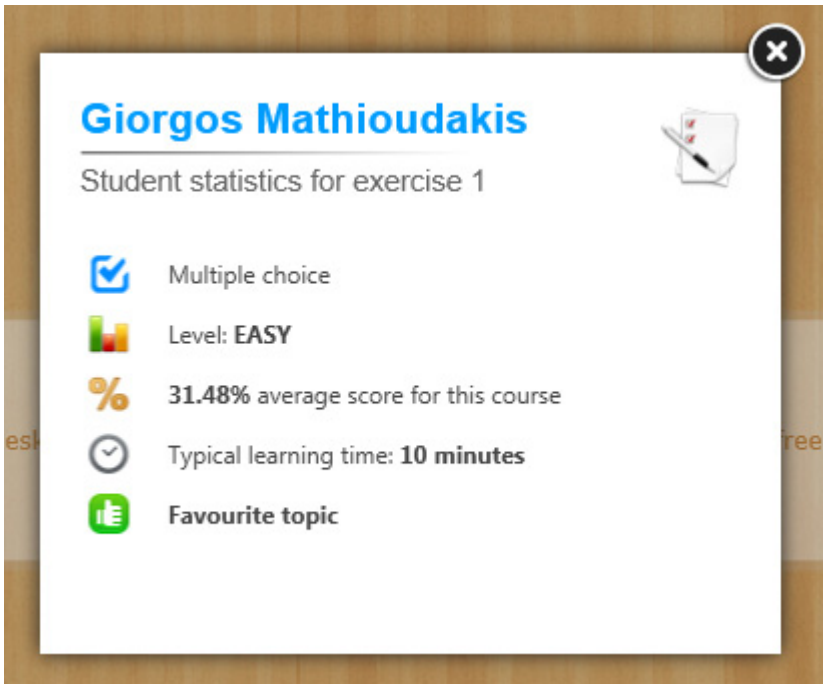


Figure 31: Exercise details and student's statistics pop-up window

5.5.3 Filtering mechanism

Managing a crowded classroom becomes a difficult task as the students' number increases. In such situations, it would be helpful for the teacher to focus on specific groups of students that require special attention. Towards this end, a filtering mechanism is implemented that aims to group the students based on their state and the type of task they are executing, and provide the tools to control the visibility of student cards belonging in these groups. The filtering mechanism is integrated in the sidebar of *Classroom Live View* and slides in from the left edge of the screen by clicking the toggle button.

The teacher can choose one or more groups that will be visible on the main area. For example, the teacher wants to focus on the active students working on an exercise. He activates the options for active students and exercise task, and de-activates all the other options. In this example, the system hides the students that have already completed the exercise or are inactive.

Figure 32 presents the sidebar while it is expanded and the options of the filtering mechanism. The filters act like checkboxes thus they are represented as ones. The available tasks to choose are study, multimedia-gallery browse and exercise.

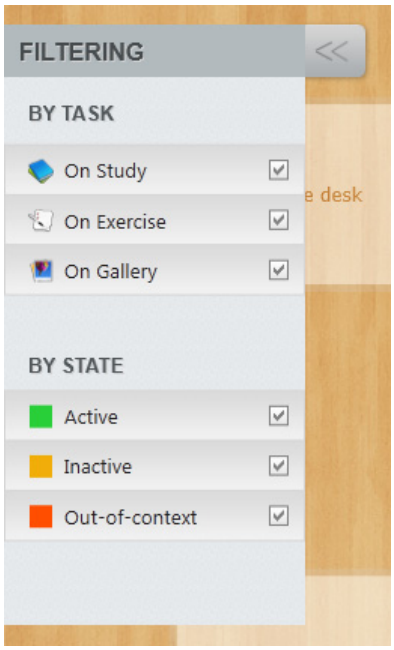


Figure 32: Filtering mechanism in sidebar (scaled)

5.5.4 Attendance

The *Attendance Record* is an official document in a typical classroom. However, in a modern and technologically augmented classroom having papers for attendance is unnecessary. When a student is identified by the classroom infrastructure to be seated at a desk, then the classroom is aware that this student is present. In addition, every classroom has a list with the registered students provided by the school manager. Therefore, all the necessary information is available to build an electronic attendance record.

The *History Logger* component records the attending students every day. If a student is not present in the classroom one school day then he is considered absent. This information is stored in the history of the classroom and is retrieved by the *Statistics* component that will be described later, to present to

the teacher how many days the student was absent. Additionally, a pop-up window shows the absent students to the teacher in real-time. When a student is identified to be in the classroom, then the system checks that this student belongs in this classroom and removes him/her from the absent list. Figure 33 presents the related window.

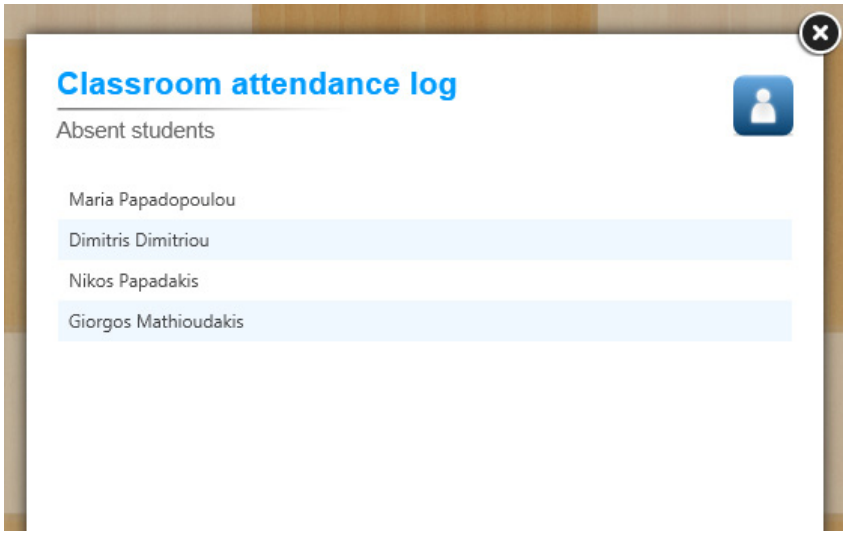


Figure 33: The attendance log (scaled)

5.6 Tests and Assignments

Tests and assignments are typical activities performed everyday in classrooms. However, in conventional classrooms these activities require a great amount of effort by the teacher in order to organized and executed properly. Towards this end, a tool is incorporated in the *Teacher Assistant* to help the teacher assign homework to the students and perform tests with ease.

5.6.1 Assignments

An assignment is a task covering an educational topic that is given to the students to work either individually or in groups in a specified time period. Assignments can have the form of an essay, a multiple-choice quiz, a fill in the

gap exercise and more. When a lesson starts, *Teacher Assistant* asks ClassMATE for a list of the available, relevant to the topic activities/tasks. The available activities are transmitted back to the *Teacher Assistant* and presented in the corresponding panel.

As presented in Figure 34 the teacher can select one task, define the deadline and set an assignment. At this point, the *Teacher Assistant* sends a message to all student desks handled by ClassMATE and informs them about the new assignment. The assignment is then stored on the student's profile and when the time specified on the deadline arrives, the student desks will ask for the assignment submission. The teacher can withdraw an assignment at any time.

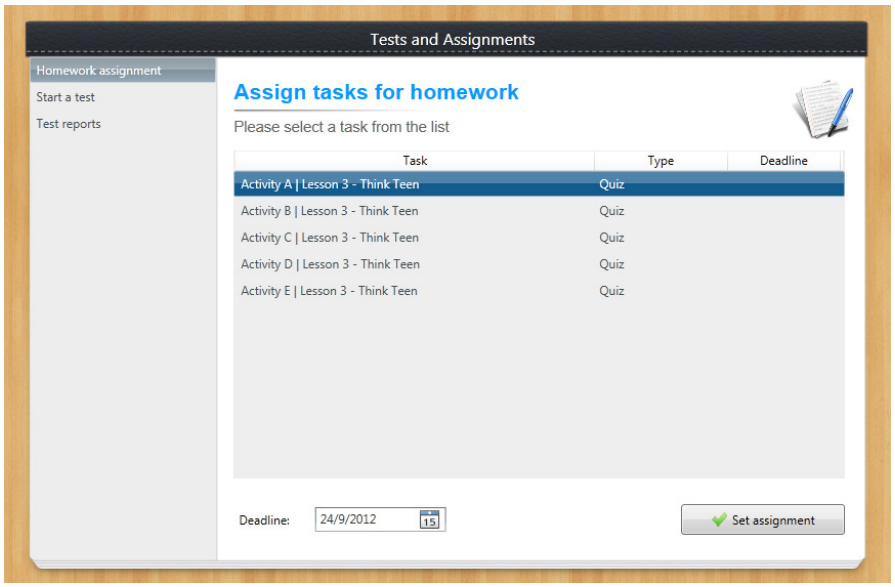


Figure 34: The homework assignments window

5.6.2 Tests

There is intuitive appeal in using assessment to support instruction [46]. Many schools formally test students at the end of a marking period, however the information from such tests is hard to use. In a formal test only a small amount of time can be allotted to each skill covered during the marking period.

Consequently, these tests are better for monitoring overall levels of achievement than for diagnosing weaknesses. Furthermore, the information of the tests arrives too late to be useful. The results can be used to make broad adjustments to curriculum but if educators are serious about using assessment to improve instruction, then more frequent and fine-grained assessments are needed. The information from such assessments can be used to modify instruction during the teaching. The teacher who consciously uses assessments to support learning takes in this information, analyzes it and makes instructional decisions that address the understandings and misunderstandings that these assessments reveal.

The AMI-RIA system incorporates functionality for assessment by providing a mechanism to the teacher in order to assign easily tests at any time. Tests are actually a form of exercise with several constraints. During testing, however, the students are not allowed to open any books and cannot use any hints regarding the questions. Because of this similarity, it was chosen to model tests with the XML format implemented for exercises. This XML format defines the questions and the available answers (if exist). More than one exercises can be represented in a single file, thus structuring a complete test. Figure 35 presents an example file containing a multiple choice exercise.

```
<?xml version='1.0'?>
<Exercise>
    <Type>MCExercise</Type>
    <Questions>
        <Question>
            <Name>mc1</Name>
            <Number>1</Number>
            <Parts>
                <First>Newton's first Law</First>
                <Second>that for every force, there is an equal and opposite force.</Second>
            </Parts>
            <Answers>
                <Answer correct="false">tells</Answer>
                <Answer correct="false">makes</Answer>
                <Answer correct="true" >states</Answer>
                <Answer correct="false">informs</Answer>
            </Answers>
        </Question>
```

Figure 35: The exercise file structure

Every time a lesson starts, the *Teacher Assistant* asks ClassMATE for the list of the available tests or exercises. Once this list is received, the corresponding panel is populated, providing the teacher the available options. The activity list is contained in a pop-up window as it is depicted in Figure 36. The teacher may select an exercise and start the testing process. As soon as the test starts, a relevant label, presented in Figure 37 appears on top of the *Classroom Live View* panel to indicate the procedure.

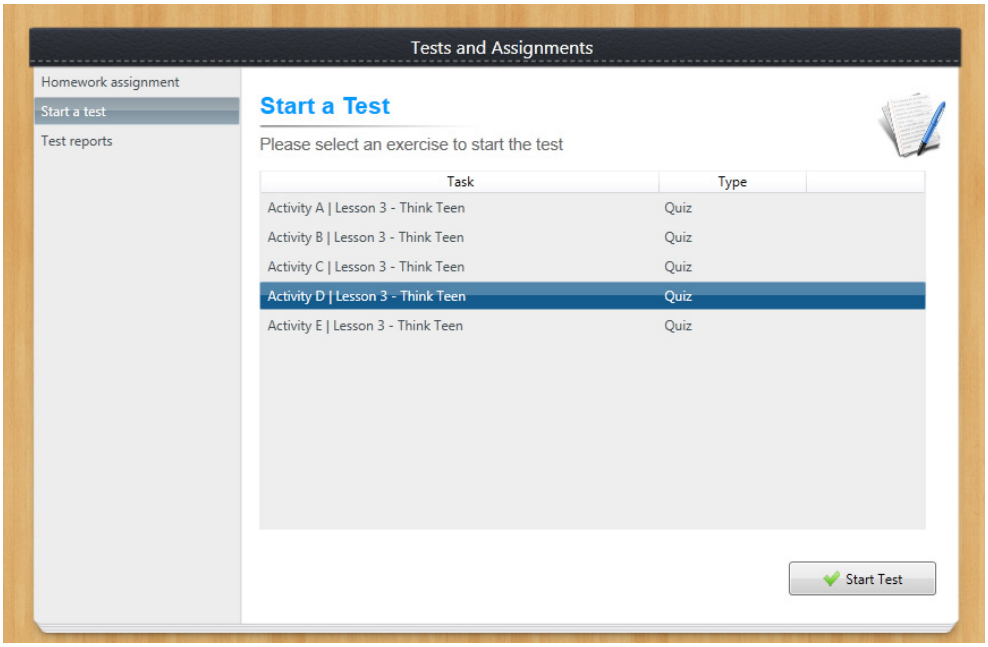


Figure 36: The tests window

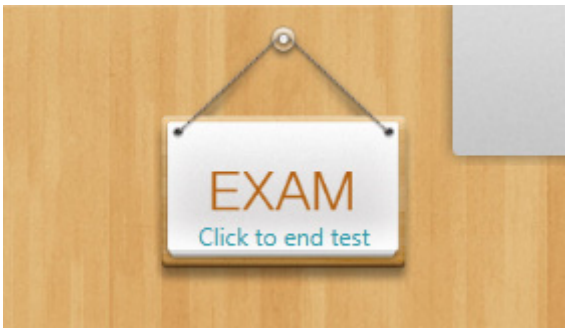


Figure 37: The label indicating the test process

When the teacher decides that the students had enough time to complete the exercises, he can force the end of the test by clicking the corresponding label appearing at top left of the window. Any tests that had not been previously submitted by the students will be automatically submitted at this point. The submitted tests are displayed to the teacher accompanied by the total score and an indicator that shows if the student had the time to submit the test or it was submitted automatically. Furthermore, the test results window displays the number of students who participated in the test and their average score. Figure 38 presents the pop-up window that displays the total scores after the test.

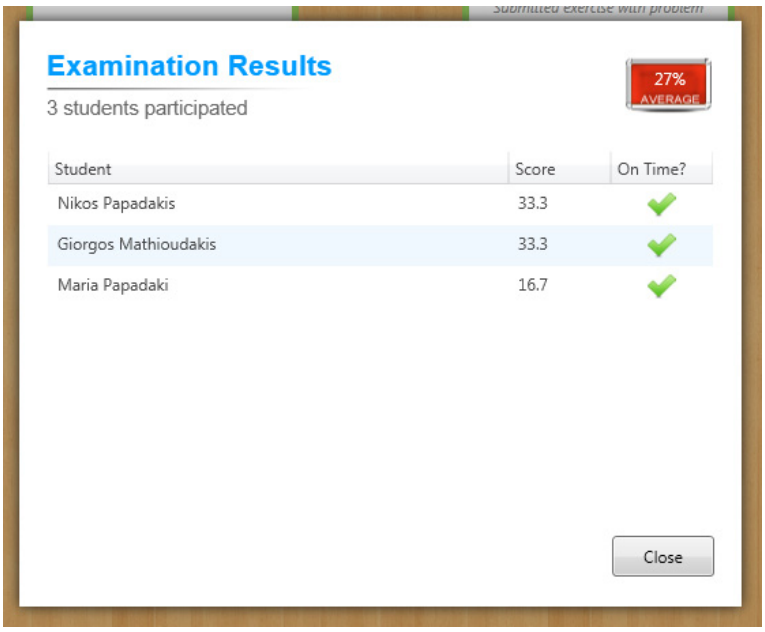


Figure 38: Examination results pop-up window

Finally, the teacher can go through past tests and check the performance of the students. The test reports panel (Figure 39) contains a list with reports from tests performed in this lesson. Every test report item is accompanied with the date that it took place. Furthermore, the teacher can select a report and watch the scores for each individual student.

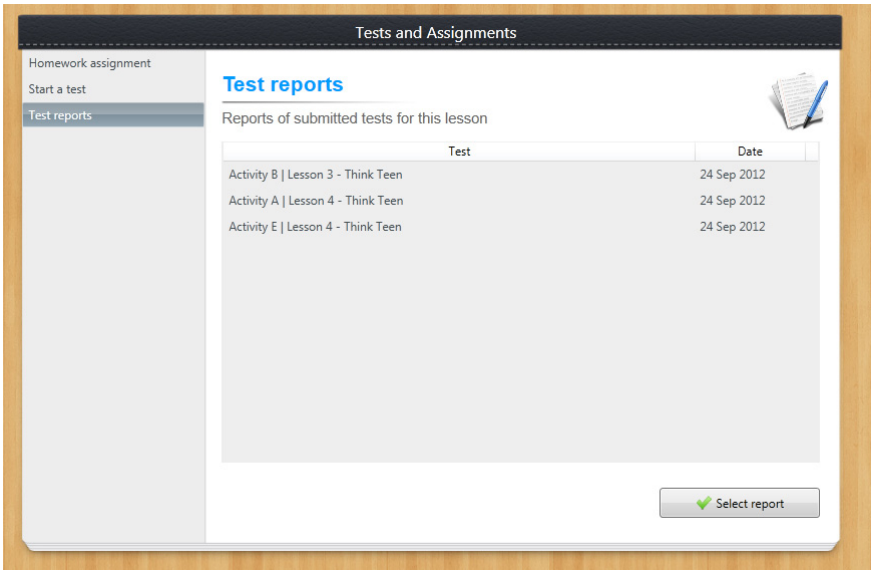


Figure 39: The test reports panel

5.7 The short-term reasoner

A teacher in the envisioned intelligent classroom is notified in real-time about the activities carried out by the students. However, sometimes the teacher will eventually miss some notifications for a student. Additionally, he/she cannot always remember the past notifications generated for one student. Therefore, there is a need of a mechanism that will discover trends in the student's activities and warn the teacher about them. For example, if a student skips one exercise this is not considered to be a big issue, however, if this student skips the fifth exercise in a row for the past few hours this means that there is a problem and the teacher should focus on this student, identify the problem and help him/her.

Furthermore, there could be more than one teacher in one classroom, e.g., one teacher for Maths and another for the English course. Thus, the one that teaches the last course during a school day is not aware of the students' states on the previous courses. In that case, a notification that one student is out of context would not be considered as alerting. However, if this particular student

is out of context for the tenth time for this day, there is something wrong and the teacher should be aware of that and try to find out what is the problem.

The short-term reasoner (STR) was implemented to address the issue discussed above. The STR is a mechanism running in the background checking the student's activities and their activity history. Every time an event about a student's activity arrives at the teacher's desk, this mechanism checks the history against this activity and retrieves the number of times this activity happened during a specified time period. If the number of times is larger than a threshold defined by the teacher, then the mechanism fires a special notification. For the information retrieval from the history, some custom SPARQL queries were implemented that take as arguments the activity entity and the time interval to check.

The STR notifications are displayed on top of the student card. A dedicated visual component slides up when an STR notification is fired. As presented in Figure 40, the visual component contains a description about the activity involved and a close button. After the teacher reads the notification, he can turn it off by hitting the close button.

Figure 40: A short term reasoner notification

However, using this feature is optional, and depends on the teacher's preferences. The teacher can define if he/she wants to use this feature and in what level. For example, one teacher would find useful to have STR notifications about out-of-context students and someone else would find good to have notifications just for students with problems in exercises. For this reason, the

preferences window presented in Figure 41 was developed, so that the teacher can customize the STR component according to his individual needs. The available preferences include the time interval that the STR uses to search in the history, the activation/de-activation of the activities and the threshold that fires the notifications for each activity. The system saves these preferences and loads them at the teacher's login. The preferences window is accessed through the corresponding button on the main menu bar at the top of the application. The layout of the window is built in a way to support future extensions regarding the teacher's settings.

The available activities for the STR are:

- Out of context
- Answer question with problem
- Submit an exercise with problem
- Exercise inactivity
- Skip exercise

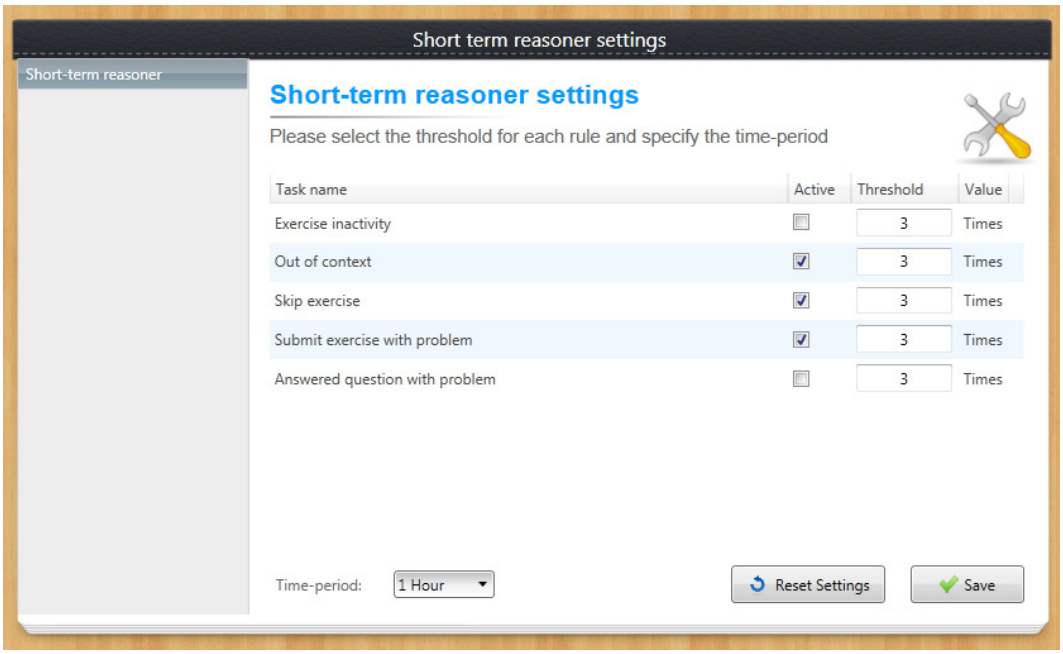


Figure 41: The STR preferences window (scaled)

5.8 Classroom activities reasoner

The *Desk Monitor's* reasoner that exists at the students' desks provides valuable insights regarding the students' activities. However, for classrooms with a large number of students, a tool is needed to observe the behaviour trends of all the students. In this context, an activity is considered to become a trend when a large percentage of students in the classroom are getting involved with it. For example, two or three students being out of context is not a big issue; instead, when twenty five students out of thirty that are in the classroom are out of context, it is a situation that requires the teacher's attention. The *Classroom Activities Reasoner* (CAR) component is developed and incorporated in the *Teacher Assistant* in order to identify the trends in the classroom and notify the teacher. To produce valuable inferences, a minimum number of students is required, initially set at 8 students. This is because having a small sample of students could produce false inferences. The following section presents the rules used by the *CAR.*

5.8.1 The Classroom Activities Reasoner rules

Out-of-context: This rule is used to identify the situation in which a large percentage of the students (>70%) are out-of-context. A classroom with the majority of students being out-of-context could mean that a break is needed or that the teacher needs to come up with something to revive his students' interest about the lesson.

Problem in exercise: When a student submits an exercise and the system identifies that he faced a problem with it, the *CAR* module checks how many of the students that worked on this exercise within a fixed period (e.g., ten hours), faced a problem too. If a large percentage (>70%) of these students indeed faced problems then this could mean that the exercise's topic should be presented again because the students have some learning gaps.

Skip exercise: When a student skips an exercise the *CAR* checks if a large part of the students started this particular exercise in a fixed period (e.g., ten hours) but skipped it at the end. When a large percentage of students (>70%) are skipping the same exercise this could mean that either the exercise is too difficult or too boring.

Working on exercise: This feature is based on the following scenario. The teacher motivates the students to start a specific exercise. The greater part of them at some time eventually will start working on this exercise. However, a small part of students for some reason may not want to work on this exercise. A rule is defined which identifies such a situation and reveals the students who do not participate in this classroom activity. The threshold for the "exercise working" trend is set to 80%. Prior to revealing the students who do not work on this exercise, the *CAR* retrieves from history and eliminates those who worked on this exercise in the past. This is because there is no obvious reason for a student to work on an exercise he/she has already completed.

5.8.2 The classroom activities reasoner notifications

When a rule fires, a relevant notification is created and displayed at the top-right part of the screen. Every notification provides a close button so that the teacher can easily discard it when it is no longer needed. If a rule fires when another notification is still placed on the screen, then it is stacked at the top of the list indicating that this is the most recent one. The notifications that appear on the screen are always chronologically sorted from the most recent (at the top) to the least recent (at the bottom). Figure 42 presents the notifications of the *Classroom Activities Reasoner*.

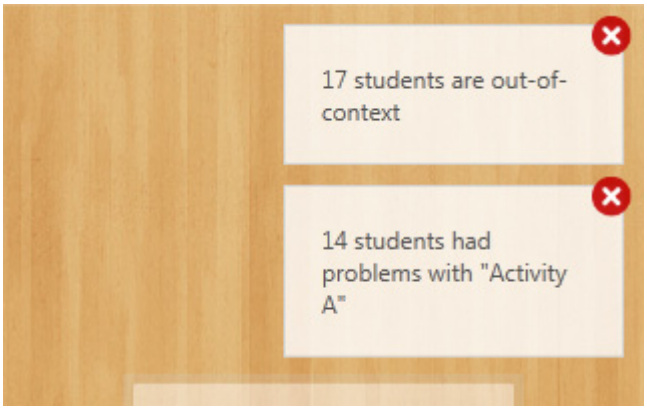


Figure 42: Notifications generated by Classroom activities reasoner

5.9 The statistics module

Within the classroom prototype described in this thesis, activities of students are collected from the augmented desks and transmitted to teacher. At the teacher's desk a rich history record is generated, which in conjunction with the defined schema constitutes a vast source of semantic information. This information can be exploited to produce valuable insights about the students' progress and performance during short or long time periods. Towards this end, a component has been developed targeted to generate graphs and statistics for the students based on the history.

The teacher can use this component to track the students' performance, their out of context time, the days that they were absent, etc. Furthermore, the teacher can identify the topics in which the students performed badly and then decide if the teaching of these topics needs to be modified. Finally, the teacher can use the generated graphs and statistics to advise the parents during their visits at school. This way, the parents can track their children's progress over time and provide help if required. The statistics component also provides a printing functionality in order to be easy to print the statistics and graphs on paper or pdf documents and hand them to the parents.

In the majority of schools, lessons are taught by different teachers. Therefore, a mechanism had to be incorporated to support the various teachers on the class. Every teacher registered in the smart classroom environment has been assigned a number of lessons. For example, John Doe is teaching algebra and trigonometry. A mechanism has been implemented to offer John the opportunity to select whether he wants to view students' statistics regarding only the lessons he is teaching or all the lessons for this classroom. By this, he can focus on his courses and discover the topics that the students are facing problems but, at the same time, he can also compare students' performance on his courses against all the other courses. Figure 43 presents the "All courses", "My courses" selection buttons.

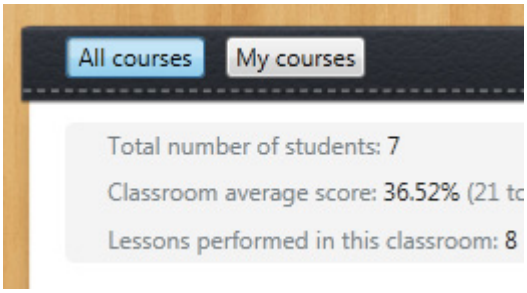


Figure 43: Statistics panel – All courses/ My courses selection

The statistics module collects data from the history record within a time period that can be defined from the teacher. The predefined set of periods includes the following options: (i) One week, (ii) Two weeks, (iii) Three weeks, (iv) One month, (v) Three months and (vi) Six months. When the teacher selects a period, the graphs and statistics are automatically regenerated to provide the corresponding data. Figure 44 presents the dropdown used for the period selection.

Two views are available for the statistics module, namely, the classroom view and the student view. The classroom view is used to display graphs and statistics for the classroom and the student view is used for individuals. The teacher can easily switch between the classroom and the student view from the

mechanism incorporated on the left sidebar. This mechanism displays the classroom view option at the top and the students' names following. When the teacher selects one of the available options, the system generates the corresponding statistics and graphs. Figure 45 presents the left sidebar that contains the mechanism.

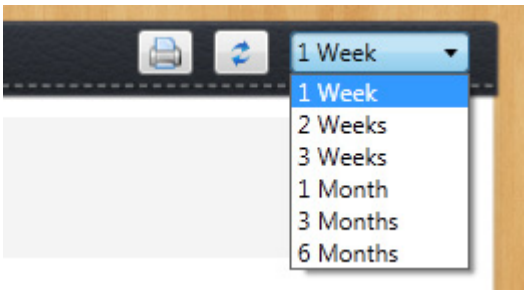


Figure 44: Statistics panel - Time period selection

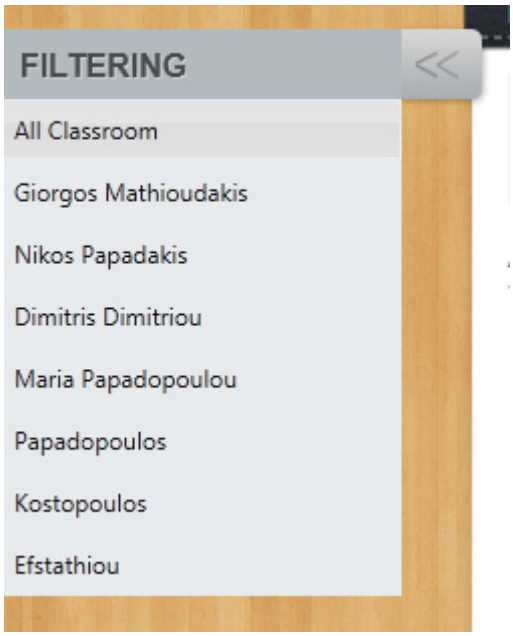


Figure 45: Student selection mechanism incorporated in left sidebar

As described previously, the classroom view includes statistics and graphs regarding all the students in the classroom. For the classroom view three statistics are defined: (i) The total number of students, (ii) the classroom's average score on exercises/tests along with the total number of submissions

and (iii) the number of lessons performed in this classroom. This information is utilized to provide the teacher an overview of the classroom. Figure 46 presents the statistics part of the classroom view.

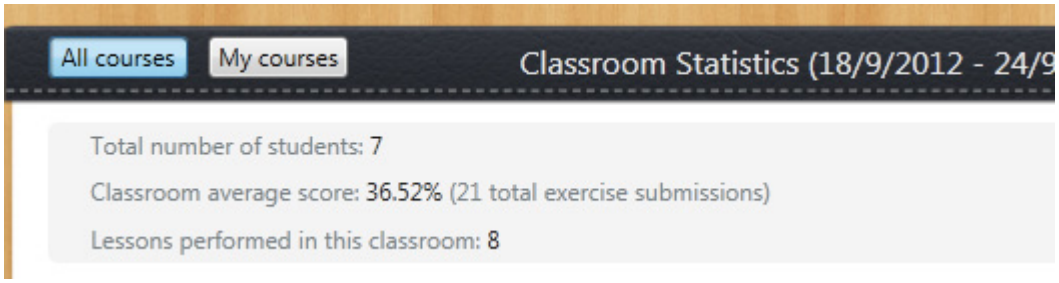


Figure 46: Classroom statistics (scaled)

Additionaly, a set of six bar charts are defined and presented in Figure 47. In particular:

Average score per day: This bar chart illustrates the average score per day the students performed over the given period.

Average out-of-context time per day: This bar illustrates the average out-of-context time per day for the total number of students.

Higher score topics: This bar chart illustrates the topics that the students had their greatest scores, along with the course that the topic belongs to.

Lower score topics: This bar chart illustrates the topics that the students had their lowest scores.

Students ranking: This bar chart presents all the students sorted by their average performance on exercises during the given time period. At the top of the list exists the student with the highest score and at the bottom of the list exist the student with the lowest score.

Figure 47: Classroom statistics and bar charts (scaled)

The student view provides statistics and graphs about the student's individual performance. The statistics part include: (i) the average score along with the number of completed exercises, (ii) the average out-of context time per day and (iii) the days he/she was not present in the classroom during the specified period. In addition, six bar charts are defined for the student view and presented in Figure 48. In particular:

Average score per day: This bar chart illustrates the average score per day of this student during the given period. If the student did not complete any exercises or tests for a day, the bar will look collapsed.

Out of context time per day: This bar illustrates the total out of context time of the student per day.

Higher score exercises (categorized by topic): This bar chart presents the exercises with the higher scores categorized by topic e.g. gravity

Lower score exercises (categorized by topic): This bar chart presents the exercises with the lower scores categorized by topic e.g. Grammar

Lesson ranking: This bar chart illustrates the performance of this student at a lesson basis

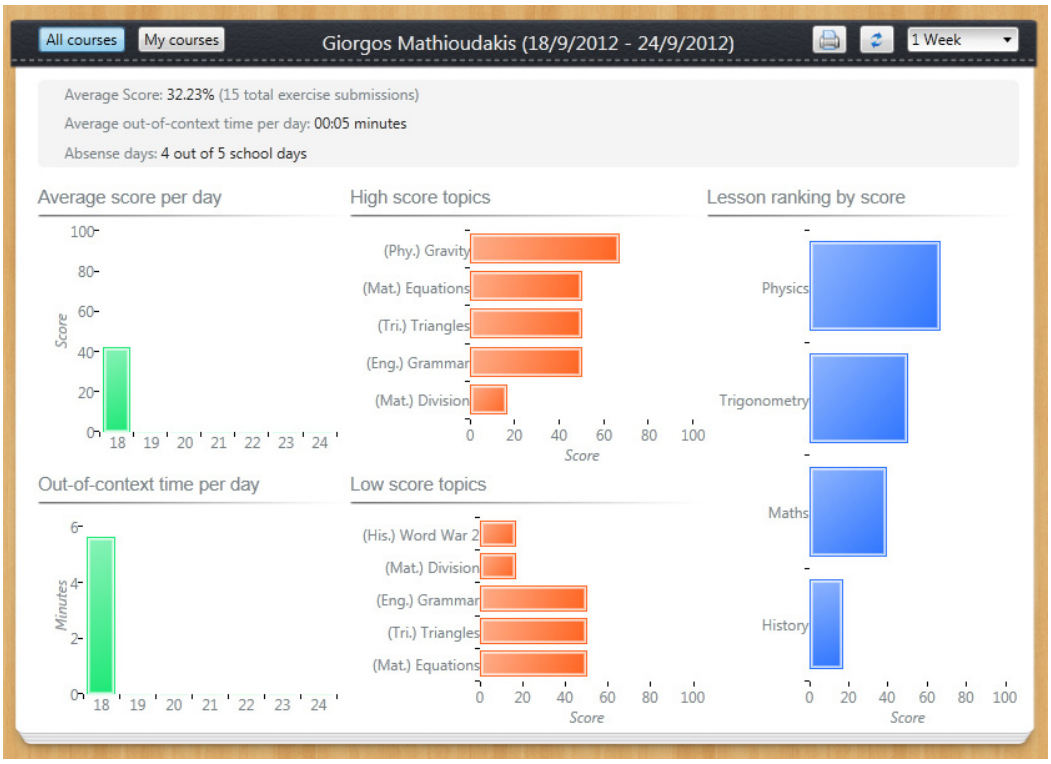


Figure 48: Student statistics and bar charts (scaled)

5.9.1 Threading & Caching

The statistics and graphs provided by the statistics component should be re-generated each time the teacher changes the period of time, the "all courses" – "my courses" option or selects a different student from the list. However, both the execution of SPARQL queries on the data-model and the generation of the graphs by the WPF toolkit[9] require fair amounts of computational power. In

[9] http://wpf.codeplex.com/

order to boost the speed of the application, a mechanism was developed that utilizes threading and the dynamic memory to provide fast and asynchronous responses.

Every time the teacher selects a new parameter, the system starts a new thread responsible to carry out the process of generating the graphs. The user-interface updates instantly and the data are loaded asynchronously when ready. The second performance tweak on the graphs regeneration involves a custom caching mechanism. This mechanism is used to store in the dynamic memory the generated graphs for future use within a session. When the teacher wants to see an instance of the statistics that was previously generated with the same parameters, a cached version of this instance is served. This prevents the system from regenerating the already generated statistics. If the teacher requests statistics for a student that are not cached, the system generates them and keeps them in memory.

5.10 The classroom schedule

Another important component of the classroom is the lessons' schedule. In a conventional classroom, the schedule is created by the manager, printed in paper and distributed to the teachers. However, in a technologically augmented classroom there is no need for paper schedules. The course schedule should be available at the same time the manager creates it. Towards this end, entities for the courses and the school days and hours are incorporated in the implemented *Classroom* ontology. The system loads the lesson schedule data from the data-model and creates a visual representation of it. The schedule view (Figure 49) displays a weekly view and it offers buttons for browsing through the weeks. In addition, the schedule view marks the current day to be easier for the teacher to identify the courses following.

Furthermore, the task assignments are displayed in the lesson schedule. A usage scenario revealing the contribution of this feature is the following. A teacher wants to assign a task for the next day. After a quick view on the lesson

schedule, he discovers that there are two more assignments for that same day. Because the students will be already busy doing these tasks, he decides to give some additional time for this assignment and schedule it two days later. An assignment is represented by a notepad icon next to the corresponding lesson. A mini legend exists at the bottom of the schedule window to define the meaning the icon.

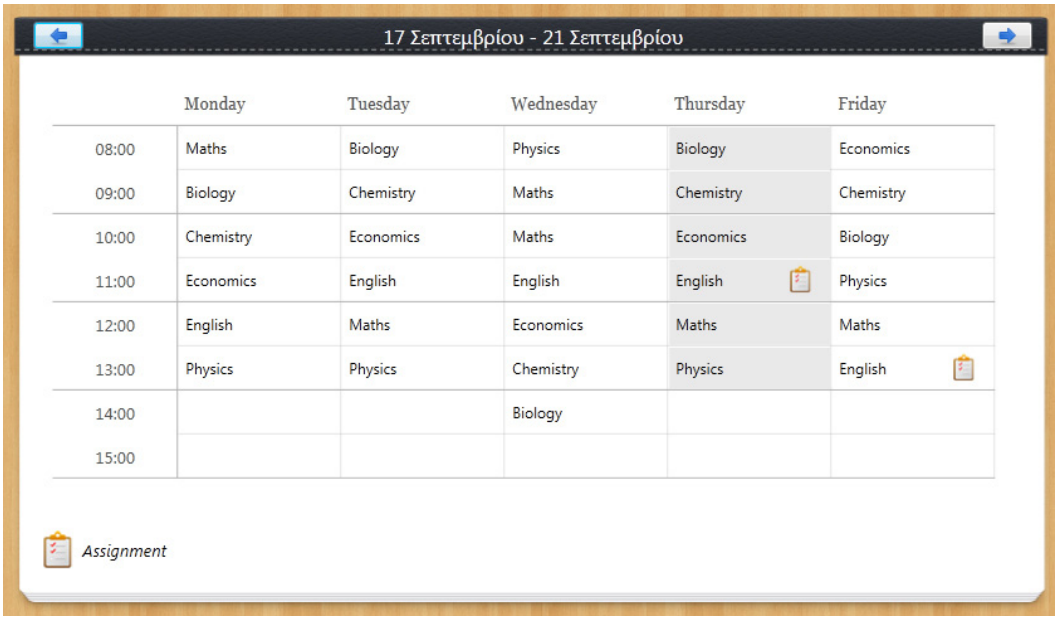


Figure 49: Weekly schedule view

Evaluation

As a first step towards the evaluation of the AMI-RIA system, a heuristic evaluation was conducted in order to identify usability errors regarding the *Teacher Assistant* application. Heuristic evaluation is the most popular usability inspection method and is carried out as a systematic inspection of a user interface design for usability [47]. It is targeted to find usability problems in the design so that they can be attended to as part of an iterative design process. The heuristic evaluation procedure involves having a small set of evaluators examine the interface and judge its compliance with a set of recognized usability principles, called heuristics. In general, heuristic evaluation should involve more than a single evaluator, since experience has shown that different people find different usability problems. The optimal approach, according to Nielsen [48], is to involve three to five evaluators, since larger numbers do not provide much additional information.

In heuristic evaluation each individual evaluator inspects the interface alone and compares it with a list of recognized usability principles, the heuristics (Table 3). Only after all evaluations have been completed, the evaluators are allowed to communicate and have their findings aggregated. This procedure is important in order to ensure independent and unbiased evaluations from each evaluator. An observer can be used to assist the evaluators in operating the system in case of problems and write down the evaluators' notes, reducing the workload of them. The results of the evaluation are available fairly soon after the last evaluation session since the observer only needs to understand and organize one set of personal notes, not a set of reports written by others.

Finally, each evaluator is called to provide severity ratings on the usability errors appearing on the aggregated list, independently of the other evaluators. The severity ratings range from zero to four and are used to indicate how

serious and how important is to fix it. Table 4 presents the available severity ratings and their meaning.

#	Heuristic	Meaning
1	Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time
2	Match between system and the real world	The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order
3	User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo
4	Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions
5	Error prevention	Even better than good error messages is a careful design which prevents a problem from occurring in the first place
6	Recognition rather than recall	Minimize the user's memory load by making objects, actions and options visible. The user should not have to remember information from one part of the dialogue to another. Instruction for use of the system should be visible or easily retrievable whenever appropriate
7	Flexibility and efficiency of use	Accelerators, unseen by the novice user, may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions
8	Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the

		relevant units of information and diminishes their relative visibility
9	Help users recognize, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution
10	Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Table 3: The usability heuristics

Severity rating	Meaning
0	I don't agree that this is a usability problem at all
1	Cosmetic problem only: need not be fixed unless extra time is available on project
2	Minor usability problem: fixing this should be given low priority
3	Major usability problem: important to fix, so should be given high priority
4	Usability catastrophe: imperative to fix this before product can be released

Table 4: Severity ratings

The evaluation of the *Teacher Assistant* was conducted by 4 evaluators, following the aforementioned process. The usability issues identified by the evaluators were noted down by an observer, who was monitoring the evaluation progress. The complete list of usability issues along with the violated rules and the average values of the severity ratings are presented in Table 5.

Issue #1

Description: The selected item on the main menu is not easily identifiable

Rules violated: Visibility of system status, Recognition rather than recall

Severity rating: 2.75

Issue #2

Description: There are no descriptive labels for the items of the main menu

Rules violated: Recognition rather than recall

Severity rating: 2.25

Issue #3

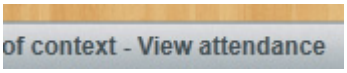
Description: The menu items look inactive due to the lack of colour

Rules violated: Consistency and standards

Severity rating: 2.5

Issue #4

Description: The attendance access link on the footer is not easily located

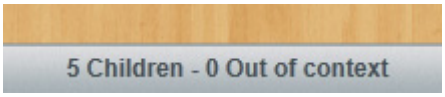


Rules violated: Recognition rather than recall, Match between the system and the real world

Severity rating: 3.25

Issue #5

Description: The information appearing on the footer bar lacks of colour contrast and it is difficult to read

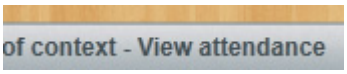


Rules violated: Aesthetic and minimalist design, Visibility of system status

Severity rating: 1.75

Issue #6

Description: The attendance link on the footer bar is not easily pressed when working on a touch screen due to the size of the bar



Rules violated: Flexibility and efficiency of use

Severity rating: 2.75

Issue #7

Description: The items on the filtering sidebar act as checkboxes but they are not identified as ones

Rules violated: Match between system and the real world

Severity rating: 2.75

Issue #8

Description: The items on the filtering sidebar cannot be easily pressed due to the lack of size

Rules violated: Match between system and the real world

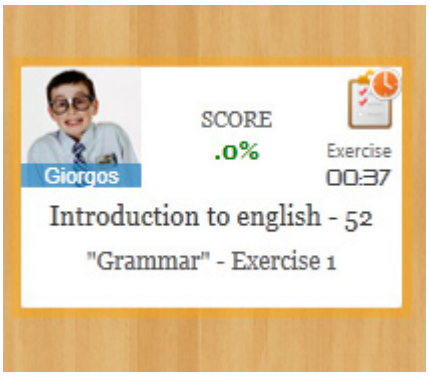
Severity rating: 3

Issue #9

Description: Clickable items on a student card are not easily identifiable

Rules violated: Recognition rather than recall

Severity rating: 3

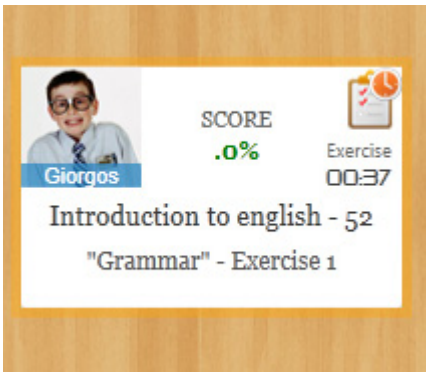


Issue #10

Description: When a student appears to be in inactivity states the task description is not updated accordingly (e.g. Exercise inactivity instead of exercise)

Rules violated: Recognition rather than recall

Severity rating: 1.5

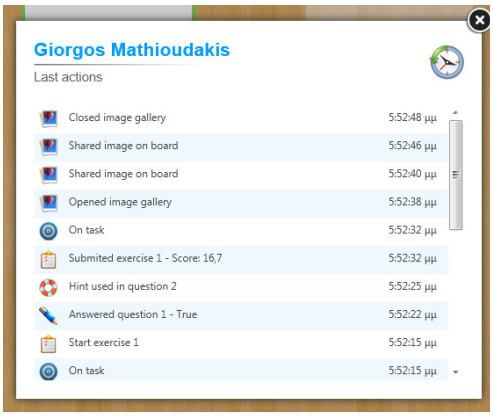


Issue #11

Description: There is no direct access for an exercise overview from the latest actions window

Rules violated: Flexibility and efficiency of use

Severity rating: 2.5

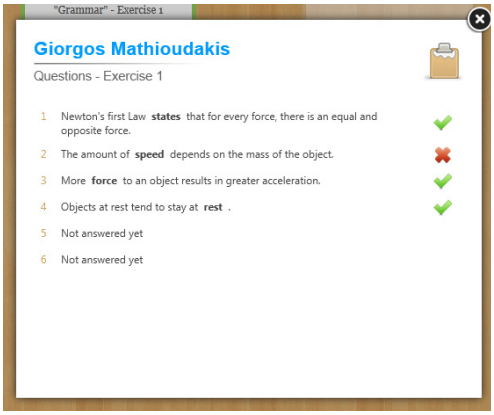


Issue #12

Description: There is no direct access from the questions window for the hints number a student used during an exercise

Rules violated: Flexibility and efficiency of use

Severity rating: 2.37

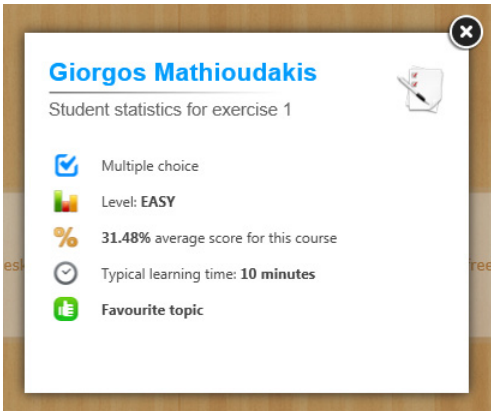


Issue #13

Description: There is no direct way to switch between the exercise questions window and the exercise details window

Rules violated: Flexibility and efficiency of use

Severity rating: 2.5

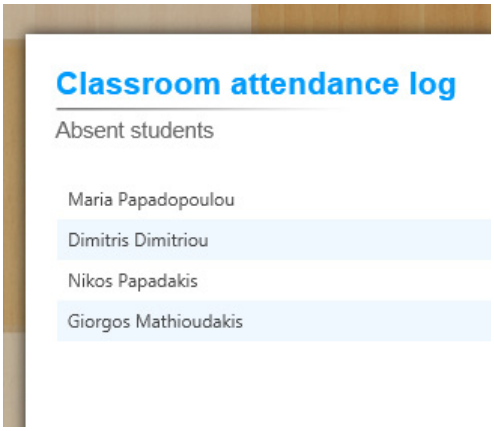


Issue #14

Description: The photos of the absent students are missing from the attendance record

Rules violated: Recognition rather than recall

Severity rating: 1.87



Issue #15

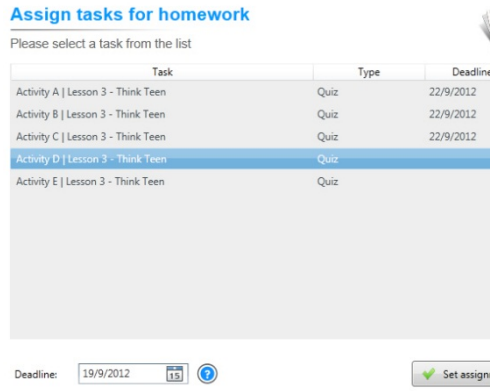
Description: Windows default datepicker cannot be handled easily on a touch screen

Rules violated: Flexibility and efficiency of use

Severity rating: 2.9

Issue #16

Description: Colour contrast of the selected item on the task assignment panel is not sufficient

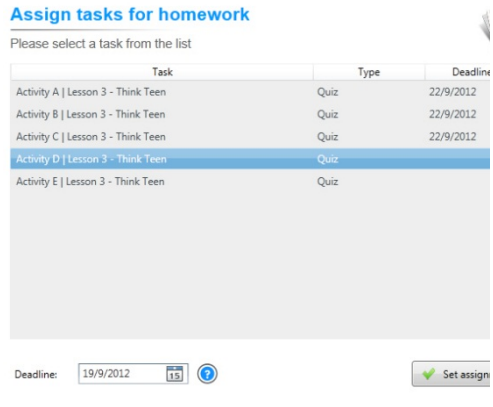


Rules violated: Aesthetic and minimalist design

Severity rating: 1.87

Issue #17

Description: When an assignment is set by the teacher no success message appears



Rules violated: Error prevention, Visibility of system status

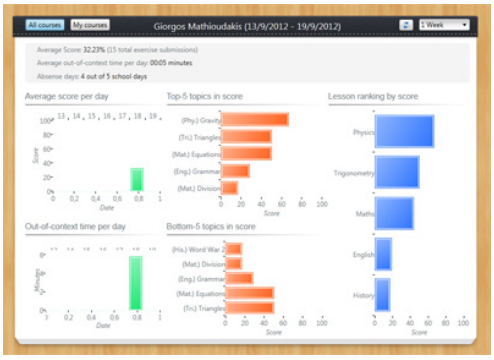
Severity rating: 2.6

Issue #18

Description: The user-interface on the statistics window displays too much information (it is cluttered)

Rules violated: Aesthetic and minimalist design

Severity rating: 2.6

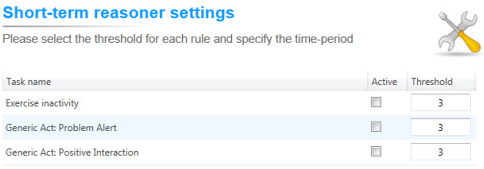


Issue #19

Description: There is no descriptive label about the threshold values

Rules violated: Error prevention

Severity rating: 2.8

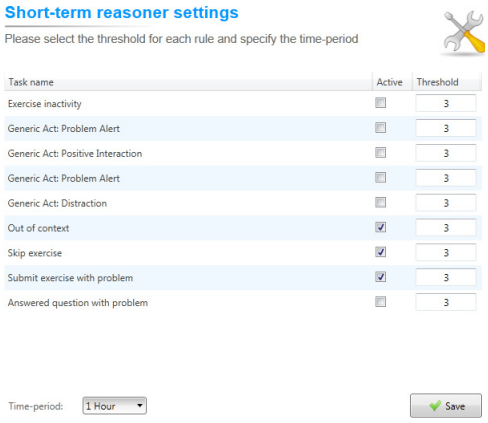


Issue #20

Description: There is no reset (undo) button for the settings

Rules violated: Help users recognize, diagnose, and recover from errors

Severity rating: 2.75



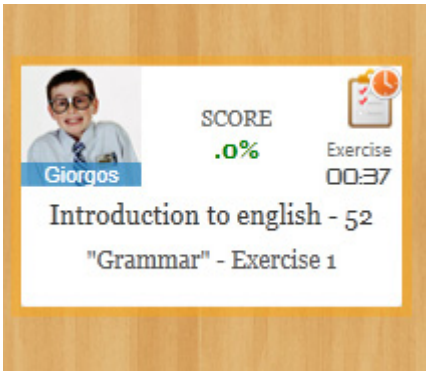
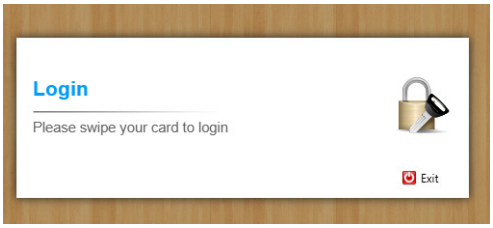
Issue #21	
Description: Information displayed on the student card and disappear with timeouts (after an amount of time) confuses the user **Rules violated**: Visibility of system status **Severity rating**: 3.25	
Issue #22	
Description: The placement of the "exit application" link may mislead the user to click on it to hide the login screen **Rules violated**: Error prevention **Severity rating**: 1.87	

Table 5: Complete list of usability issues discovered during the heuristic evaluation process

The evaluation process identified 11 major usability issues with severity score above 2.5 and 11 minor issues below 2.5. Prior to the launch of the first prototype of the *Teacher Assistant,* there was an effort to eliminate the majority of the issues discovered during the evaluation. As a result, fifteen out of twenty two usability problems were eliminated including ten out of eleven major issues. The rest of the problems will be addressed in a later version of the system before any user-based evaluation takes place.

A user-based evaluation of the AMI-RIA system in a real classroom is required to be conducted in future steps in order to acquire additional feedback from end users regarding the effectiveness and the efficiency of the system in everyday classroom activities, as well as how useful teachers actually find the system.

Conclusion and future work

5.11 Summarizing

This thesis presented a real-time system targeted to assisting the teacher within the context of the intelligent classroom. The proposed system monitors the students' activities in an unobtrusive way and generates valuable insights in order to assist the teacher keep track of individual students. Thereby, the teacher is supplied with the needed information to decide when and how to adapt the teaching strategy.

The AMI-RIA system presented in this thesis provides a significant contribution to the smart classroom, as it addresses the teacher's perspective. The backbone infrastructure of the classroom named ClassMATE, in collaboration with several artifacts creates the conditions needed for a data collection mechanism.

The system is based on a modular architecture. The first major part of the system, named *Desk Monitor*, is an agent type module deployed in every student desk. This module in collaboration with ClassMATE collects data regarding the students' activities and stores them as RDF triples. A proofing mechanism exploits the collected data, the defined taxonomies and a set of rules in order to identify situations that require the teacher's attention and intervention. The data and the generated inferences are transmitted to the teacher's desk where the second part of the system named *Teacher Assistant* is located.

The *Teacher Assistant* aims to present in real-time an overview of the classroom by exploiting the information originating from the students' desks. A friendly user-interface is used to help the teacher keep control and manage the classroom effectively. In addition, two components are incorporated to the *Teacher Assistant* to enhance the monitoring process. The *Short Term Reasoner*

is used to discover repeatable activities in short time periods and the *Classroom Activities Reasoner* is used to identify trends of activities in the classroom.

Additionally, a component named *History Logger* makes use of the collected data to create a semantic history record based on the defined *classroom* ontology. A statistics component is incorporated in the system and used to perform queries against the history record, in order to discover and retrieve information regarding the students' progress and performance. Furthermore, the statistics component aims to discover topics that need adaptation.

Finally, a set of tools have been developed and deployed on the teacher's desk targeted to enhance typical procedures that can be found in conventional classrooms such the attendance record, the lessons schedule and the homework assignments.

5.12 Future work

The realized evaluation of this work provided some useful insights regarding the usability of the system. The feedback gained by the evaluators will be used as a base for the next release of the AMI-RIA system. Some example issues noted by the evaluators include simplifying the interface of the statistics component and provide more interconnections between the various components in the *Teacher Assistant* application.

The next step of this work would be to conduct a full-scale user-based evaluation of the system in a real classroom in order to acquire additional feedback from end users regarding the effectiveness of the system. The evaluation experiment will include real students and a teacher in the context of a typical school day. The results will lead to improvements on the currently implemented rules for identifying the special situations of the students.

Additionally, some relevant topics were investigated for future upgrades. The real-time assisting tools provide a great environment for the teacher to

manage the classroom. However, it is the teacher's responsibility to intervene, support or reward the students. Sometimes, the teacher would like to intervene and support the students without actually interfering or interrupting their activities. For example, if a student achieves a great score on an exercise the teacher could send him a badge, rewarding him for this achievement. Alternatively, if a student faces problems with an exercise the teacher could send him relevant educational content, thus helping him to complete the exercise and fill in any learning gaps.

Furthermore, a great addition to the system would be to make the student desks aware of any inferences produced during the reasoning process. This way the students will be informed as well about any problems occurred regarding their activities. The feedback provided by the system could be used by the students to adjust their activities accordingly. For example, when a student is off-task the system would notify him/her about the steps required to return on-task.

Another important extension of the system would be a tool for creating or adjusting the implemented Notation3 rules. This tool will offer an easy-to-use graphical environment for the teachers to adjust rules by combining the condition facts and defining one or more relevant inferences. Ideally, entities, properties and values from the data-model will be presented as graphical modules that can be dragged and dropped, building this way the new rules.

Finally, an important extension of the system would be to develop the infrastructure to aggregate the information originating from the classrooms and provide a tool for the school administration to keep track of all the students and classrooms. Ideally, this extension could potentially replace the paper reports regarding the students' progress and performance.

Bibliography

[1] J. Augusto and C. Nugent, Designing smart homes: the role of artificial intelligence, vol. 4008, Springer-Verlag New York Inc, 2006.

[2] G. Abowd, C. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney and M. Tani, "Teaching and learning as multimedia authoring: the classroom 2000 project," in *Proceedings of the fourth ACM international conference on Multimedia*, 1997.

[3] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao and F. Liu, "The smart classroom: merging technologies for seamless tele-education," *Pervasive Computing, IEEE,* vol. 2, no. 2, pp. 47-55, 2003.

[4] G. Margetis, A. Leonidis, M. Antona and C. Stephanidis, "Towards ambient intelligence in the classroom," *Universal Access in Human-Computer Interaction. Applications and Services,* pp. 577-586, 2011.

[5] R. Ramadan, H. Hagras, M. Nawito, A. Faham and B. Eldesouky, "The Intelligent Classroom: Towards an Educational Ambient Intelligence Testbed," in *Intelligent Environments (IE), 2010 Sixth International Conference on*, 2010.

[6] M. Korozi, "PUPIL- Pervasive UI develoPment for the ambIent cLassroom," 2010.

[7] C. O'Driscoll, M. Mithileash, F. Mtenzi and B. Wu, "Deploying a Context Aware Smart Classroom," 2008.

[8] M. Koutraki, V. Efthymiou and G. Antoniou, "S-CRETA: Smart Classroom Real-Time Assistance," *Ambient Intelligence-Software and Applications,* pp.

67-74, 2012.

[9] D. Pearce-Lazard, A. Poulovassilis and E. Geraniou, "The design of teacher assistance tools in an exploratory learning environment for mathematics generalisation," *Sustaining TEL: From innovation to learning and practice,* pp. 260-275, 2010.

[10] C. Murphy, G. Kaiser, K. Loveland and S. Hasan, "Retina: helping students and instructors based on observed programming activities," in *ACM SIGCSE Bulletin*, 2009.

[11] M. Rodrigo and R. Baker, "Coarse-grained detection of student frustration in an introductory programming course," in *Proceedings of the fifth international workshop on Computing education research workshop*, 2009.

[12] I. Jormanainen, Y. Zhang, E. Sutinen and others, "Agency Architecture for Teacher Intervention in Robotics Classes," in *Advanced Learning Technologies, 2006. Sixth International Conference on*, 2006.

[13] L. Soh, N. Khandaker and H. Jiang, "I-MINDS: a multiagent system for intelligent computer-supported collaborative learning and classroom management," *International Journal of Artificial Intelligence in Education,* vol. 18, no. 2, pp. 119-151, 2008.

[14] L. Soh, N. Khandaker, X. Liu and H. Jiang, "A computer-supported cooperative learning system with multiagent intelligence," in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006.

[15] L. Soh, X. Liu, X. Zhang, J. Al-Jaroodi, H. Jiang and P. Vemuri, "I-MINDS: an agent-oriented information system for applications in education," in *Agent-Oriented Information Systems*, 2004.

[16] L. Soh, H. Jiang and C. Ansorge, "Agent-based cooperative learning: a proof-

of-concept experiment," in *ACM SIGCSE Bulletin*, 2004.

[17] L. Kepka, J. Heraud, L. France, J. Marty and T. Carron, "Activity visualization and regulation in a virtual classroom," in *Proceedings of the 10th IASTED International Conference on Computers and Advanced Technology in Education*, 2007.

[18] L. France, J. Heraud, J. Marty, T. Carron and J. Heili, "Monitoring virtual classroom: Visualization techniques to observe student activities in an e-learning system," in *Advanced Learning Technologies, 2006. Sixth International Conference on*, 2006.

[19] H. Chernoff, "The use of faces to represent points in k-dimensional space graphically," *Journal of the American Statistical Association,* pp. 361-368, 1973.

[20] R. Mazza and V. Dimitrova, "Generation of graphical representations of student tracking data in course management systems," in *Information Visualisation, 2005. Proceedings. Ninth International Conference on*, 2005.

[21] R. Mazza and V. Dimitrova, "CourseVis: A graphical student monitoring tool for supporting instructors in web-based distance courses," *International Journal of Human-Computer Studies,* vol. 65, no. 2, pp. 125-139, 2007.

[22] T. Chronopoulos and I. Hatzilygeroudis, "An intelligent system for monitoring and supervising lessons in LAMS," in *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, 2010.

[23] Y. Georgalis, D. Grammenos and C. Stephanidis, "Middleware for ambient intelligence environments: Reviewing requirements and communication technologies," *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments,* pp. 168-177, 2009.

[24] M. Antona, G. Margetis, S. Ntoa, A. Leonidis, M. Korozi, G. Paparoulis and C.

Stephanidis, "Ambient Intelligence in the classroom: an augmented school desk," in *Proceedings of the 2010 AHFE International Conference (3rd International Conference on Applied Human Factors and Ergonomics), Miami, Florida, USA*, 2010.

[25] G. Margetis, X. Zabulis, P. Koutlemanis, M. Antona and C. Stephanidis, "Augmented interaction with physical books in an Ambient Intelligence learning environment," *Multimedia Tools and Applications,* pp. 1-23, 2012.

[26] O. Lassila, R. Swick and others, "Resource description framework (RDF) model and syntax specification," 1998.

[27] R. V. Guha and D. Brickley, "{RDF} Vocabulary Description Language 1.0: {RDF} Schema," 2004.

[28] J. D. ROO, *Euler proof mechanism,* 2005.

[29] T. Berners-Lee, "Notation 3-An readable language for data on the Web," 2006.

[30] E. Prud'hommeaux and A. Seaborne, "{SPARQL} Query Language for {RDF}," 2008.

[31] T. Tudorache, J. Vendetti and N. Noy, "Web-Protege: A lightweight OWL ontology Editor for the Web," *SDR].(Cit. on p.),* 2008.

[32] J. Tauberer, *SemWeb.NET: Semantic Web/RDF Library for C#/.NET,* 2010.

[33] M. Porter and others, *An algorithm for suffix stripping,* Program, 1980.

[34] D. Cooper, I. Arroyo, B. Woolf, K. Muldner, W. Burleson and R. Christopherson, "Sensors model student self concept in the classroom," *User Modeling, Adaptation, and Personalization,* pp. 30-41, 2009.

[35] S. D'Mello, A. Graesser and R. Picard, "Toward an affect-sensitive

AutoTutor," *Intelligent Systems, IEEE,* vol. 22, no. 4, pp. 53-61, 2007.

[36] T. Dragon, I. Arroyo, B. Woolf, W. Burleson, R. el Kaliouby and H. Eydgahi, "Viewing student affect and learning through classroom observation and physical sensors," in *Intelligent Tutoring Systems*, 2008.

[37] U. RISK, "Draft Standard for Learning Object Metadata," 2002.

[38] R. Baker, A. Corbett, K. Koedinger and A. Wagner, "Off-task behavior in the cognitive tutor classroom: when students game the system," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2004.

[39] J. Carroll, "A model of school learning," *The Teachers College Record,* vol. 64, no. 8, pp. 723-723, 1963.

[40] M. Cocea, A. Hershkovitz and R. Baker, "The impact of off-task and gaming behaviors on learning: immediate or aggregate?," 2009.

[41] J. Beck, "Using response times to model student disengagement," in *Proceedings of the ITS2004 Workshop on Social and Emotional Intelligence in Learning Environments*, 2004.

[42] E. Gamma, Design patterns: elements of reusable object-oriented software, Addison-Wesley Professional, 1995.

[43] M. Storey, M. Musen, J. Silva, C. Best, N. Ernst, R. Fergerson and N. Noy, "Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Prot{\'e}g{\'e}," in *Workshop on Interactive Tools for Knowledge Capture (K-CAP-2001)*, 2001.

[44] D. Brickley and L. Miller, "FOAF vocabulary specification 0.91," 2000.

[45] S. Weibel, J. Kunze, C. Lagoze and M. Wolf, "Dublin core metadata for resource discovery," *Internet Engineering Task Force RFC,* vol. 2413, p. 222, 1998.

[46] C. Gears, "Classroom assessment: Minute by minute, day by day," *Assessment,* vol. 63, no. 3, 2005.

[47] J. Nielsen, *Heuristic Evaluation,* 2010.

[48] J. Nielsen, *How to Conduct a Heuristic Evaluation,* 2010.