AAMS

(Automatic Assignment Management System)

Group 21

Ajanth Thangavelu

Roberto Castañeda Lozano

Tony Karlsson

Love Jädergård

Requirements document

Contents

1 Preface

1.a Expected readers

If this would be a real project this document would be read by the developer team of this system and developers of systems that use this system. Other people interested in this document would be administrative personnel that have to decide on using this system in their organization or not. This is not a manual so end users would probably not be interested in this document. However, this is not a real project: it is a part of the Software Engineering course, so the expected readers are just the members of group 21 and the teachers of the course.

1.b Version history

- Version 1.0 (first version): 2008-01-04

2 Introduction

2.a Users and problems that the system solve for them

University teachers usually have to deal with courses with a very large number of students. Collecting assignments from all of them by e-mail can be very time-consuming and can even lead to mistakes made by the teacher (forgetting to open some e-mail, etc.). Furthermore, the manual processing of all assignments (reading, commenting and marking them) is a task that requires a lot of extra-concentration, thus making the teachers waste their time with organizing instead of focusing on the homework itself.

It is in the interest of universities to streamline the handling of assignments so that teachers can allocate time to other tasks than organizing files on their computers and figure out what to do with bad formatted e-mails from students. it is also important that the handling of assignments is safe so that students get the right grade. We believe that a system like ours will reduce the need for sedative and caffeine for both students and teachers.

Although our system will be designed with the needs of teachers in mind, AAMS could also be used in a more general way: not just by teachers but by anyone who needs to collect and review documents from a large group of people. For example project managers.

The solutions to some common problems that our system will solve are:

- The interaction with the e-mail server, in order to collect all the assignments, will be automated.
- The assignments will be automatically organized in a folder system to every student and course.
- Some tools will be given in order to attach comments and marks to each assignment.
- There will be the possibility of creating reports in order to make a general vision of the state of the course and relevant information.
- The communication with the RES system (used to register students in courses, add marks to their assignments, etc.) will be automated, letting the possibility of expanding the automation to other university systems opened.

2.b Main uses of the system

The main use of system is to let the university teachers manage the collection and administration of course assignments easily. The system is able to download from an e-mail account and organize the assignment deliveries, showing a clear vision of the state of the delivery to the user. The

system allows the user to add comments to the assignments, grade them, and send easily the feedback to the students. Some reports, showing clearly the general state of the course and the different deliveries, can be quickly generated. In order to make the workload lighter, the system can synchronize with the RES system. These are three usage narratives that exemplify some possible applications of our system:

- "Gunnar is a university professor at the institution for anthropology. He is back at his office feeling a bit dizzy. He just returned from a party held for one of his colleagues that is going to central Sahara for six months, studying ceremonial dances for young females. Gunnar is very jealous of his colleague. He decides to do some work to clear his mind from the dark thoughts. Yesterday was the last day for the students to submit their latest homework assignment. Gunnar starts AAMS (Automatic Assignment Management System) and the program downloads the students assignments from the e-mail server, presenting him with a list of all the students that have submitted their work. He cannot focus on the list, the letters are just dancing in front of him. He decides he is too drunk to work and that he would be better off if he got some sleep. He thinks it would be better if he stays the night at the office, his wife would probably agree with him. In the morning, after a quick "shower" using wet tissues, he resumes work. He starts AAMS and some late assignments drop in. He starts to work through the list, reviewing and commenting all students. His comments are automatically sent back to the students. Finally he instructs the program to update the university database using the RES system. Gunnar likes AAMS because it provides him with the tools he needs to manage the students homework assignments in a consistent way. And there is no risk of loosing any student work no matter how drunk he is."

- "Stina-Britta is a student at the Center for Gender Studies, today she is going to submit her paper on Early feminism in northern Finland. She is late, she has too many boyfriends and too little time. The teacher must have her paper before 18.00. She prepares an e-mail to her teacher. In the subject line she writes her name, the code for the course and the name of the assignment. She attaches a pdf-file, containing her work, to the e-mail. 17:30 she pushes the send button and says a prayer that her e-mail will arrive before 18:00. 18:15 she receives a confirmation e-mail from the AAMS. It tells her that her teacher received her paper at 17:58. Stina-Britta is relieved and to calm down she decides to go and watch a movie with her fifth boyfriend."

- "John Doe works as a project manager for a consulting company. John is the middleman between clients and consults. Neither John nor the consults has the time to have face to face meetings so often, therefore he once a week receives email reports from his consults.

  John uses Automatic Assignment Manage System (AAMS) to organize and manage all reports for each project the consults has been assigned. AAMS makes sure John does not miss to handle any report and allows him to give individual feedback to the consults."

2.c Context/environment in which the system is to be used

AAMS is a stand-alone application that will run on the teachers personal computer. It is not a service that runs on a server. The supported operating systems will be Windows XP, GNU/Linux (Ubuntu 7.x) and FreeBSD 6.2. It will probably run under a broader range of Linux-like systems. The hardware needed will be any personal computer that can run any of the supported operating systems. AAMS requires a connection to the Internet and an e-mail account supporting SMTP (Simple Mail Transfer Protocol) and POP3 (Post Office Protocol version 3).

The software system will be used in environments like university, office or other environments where someone have to handle electronic documents submitted by a large group of people, giving them feedback from these.

2.d Scope of the system

Topic	In	Out
Importing students information from student list files	x	
Importing homework assignments from an e-mail account	x	
Editing comments and marking assignments	x	
Sending results and comments to the students by e-mail	x	
Producing reports about the state of the course or the grades for a specific student or assignment	x	
Calculating statistics and mark averages		x
Exporting marks to student grades files	x	
Importing and exporting information from the RES system through intermediate files	x	
Supporting shared management for several teachers		x
Graphical user interface	x	
Secure e-mail communication	x	
Password protection for locally stored information		x
Cross-platform compatibility (Windows and some UNIX based operative systems)	x	
Complete e-mail client functionality		x

Figure 1: Scope of the system

In principle, our system will be able to import lists of students in order to define courses. These lists could be made manually or extracted from the specific course management system that every institution usually has. In our system, we will just develop the link with the RES system, letting it opened to future additions. Likewise, the system will be able to export lists of marks for assignments: the link with the RES system will be developed as well. The statistics and mark averages calculations are too particular for each course, so we have decided to let them out of the scope of our system. However, some general information will be provided about the status of each delivery, etc.

Another reasonable feature that will be out of the boundaries of our system is the shared management of a course. A course managed by several teachers sharing the evaluation of assignments could be emulated, for example, splitting the course into several virtual courses, one for each teacher; so this functionality, although could be useful in some cases, is not completely indispensable in the context in which the system is used.

Last, we have decided not to provide encryption for the local stored information that the system will manage, because the application is mostly intended for personal computing environments, without public access.

2.e Main factors

- If the system receives the expected assignment from a student, a receipt must be sent back to her. If something is wrong (for example, there is some required file missing), the system will automatically send a notification to the student. It is important, in general, to ensure that the students can receive all the needed feedback from the teacher (related to the assignments) without the help of an external application.
- The teacher must be able to use his or her usual e-mail account to receive the assignments in. The system will only react to the expected homework e-mails (recognizing them by the content of the "subject" field). However, it is suggested to work with a dedicated e-mail account in order to prevent coordination errors (e.g., an assignment e-mail that is accidentally marked as "read" with an external e-mail client).
- The system must include some flexibility in order to manage usual "exceptions" in the assignments collection task (e.g., if a student gives his or her homework to the teacher by hand).
- Security features will be implemented in the e-mail communication and in the communication with the RES system.
- The user interface shall be clean and easy to use (e.g presenting previous e-mails in case of a follow up from an incomplete assignment instead of letting the user of the system search manually). As the aim of the system is to save time for the user, the user interface must be also focused in this way.
- The data will be stored locally, combining a folder structure with a XML (eXtensible Markup Language) file system.
- The system must be cross-platform compatible, in order to do not limit the number of users by this reason: it must compile and run in some of the most currently representative operative systems.
- The system must be usable enough to allow a user with the minimal computer user knowledges use it without requiring too much learning time.

2.f Technologies and Risks

The software will be written in the programming language C++. C++ is a widely used programming language supported by most platforms. The familiarity of the C++ language in the group is good, therefore the risk of using the language itself is small. A possible risk is the group's lack of familiarity with cross-platform development.

wxWidgets is going to be used to develop the GUI (Graphical User Interface). WxWidgets is an API (Application Programming Interface) that is used to write cross-platform GUI applications. The supported platforms are Windows and some Unix-based operative systems. A possible risk is the group's lack of knowledge in developing wxWidgets software. And since the system is going to be used on different platforms another possible risk is the group's lack of knowledge in cross platform development with GUI applications.

For the integrated email-client in our system we are going to use an open source email-library or parts of an existing open source email-client. The protocols we are going to use are POP3 (Post Office Protocol version 3) for retrieving emails of the server and the SMTP (Simple Mail Transfer Protocol) for sending emails to the server. Both of these protocols are going to use secure transfer using an encryption protocol. A possible risk for using a library or parts of an existing client is that we do not know how well the code is written, which could affect the stability of the entire system.

We will use the XML (eXtensible Markup Language) standard to store information used in the system. XML is a widely used standard for storing of information on files. For implementing this into our system we will use an open source library. Since the know-how exists in the group there is a low risk using this technology.

Our system is going to be able to communicate with the RES system. The RES system is a system used at KTH for handling the grading of student's homework assignments and exams. A possible risk is the limit in testing the communication between our system and the RES system because the files used to store the information (the students and their grades for homework assignments and exams) in the RES system have restricted access.

3 Glossary

Glossary

API: Application Programming Interface, source code interface that an operating system or library provides to support requests for services to be made of it by computer programs. 8

assignment: course work with a defined deadline. The system assumes that the assignment must be handed in electronic format, corresponding to a single student. 5, 13–15, 20–24, 27, 31–36, 38, 42

attachment: in the system context, file that is included in an outgoing e-mail. 5, 13, 16, 22, 24, 25, 34, 36

e-mail account: in the system context, e-mail address that identifies a location to which e-mail messages can be delivered, usually provided to the user by a company or institution (e.g. the university of the teacher). 5–8, 14, 21–23, 25, 32, 36, 37

C++: general-purpose, imperative programming language, regarded as a mid-level language, as it comprises a combination of both high-level and low-level language features. 8

course: in the system context, a set of assignments related to some students and a professor. 5–7, 13, 14, 16, 19–21, 25, 28, 31, 37, 39, 42

e-mail client: In the system context, front-end computer program used to manage e-mail. 7–9, 14, 24, 34

e-mail server: computer running a program that transfers electronic mail messages from one computer to another. 5, 6, 9, 14, 16, 32, 36

e-mail (electronic mail): method of composing, sending, storing, and receiving messages over electronic communication systems. In the context of this document we will always refer to it as the Internet based system. 5, 14, 16, 21, 24, 32, 36, 42

subject line: field of the header of an e-mail that contains a brief summary of the contents of the message. 6, 8, 13, 32, 43

FreeBSD 6.x: Unix-like free operating system descended from UNIX via the Berkeley Software Distribution. 6, 18

feedback: in the system context, information about an assignment that is given back from the professor to the student. 6–8, 16, 21, 22, 36, 37

GNU/Linux Ubuntu 7.x: predominantly desktop-oriented Linux distribution based on Debian GNU/Linux. 6, 18

GPL: General Public License, widely used free software license, originally written for the GNU project. It is a strong copyleft license that requires derived works to be available under the same copyleft. 18

GUI: Graphical User Interface, type of user interface which allows people to interact with a computer presenting graphical icons, visual indicators or special graphical elements called "widgets", instead of offering only text menus, or requiring typed commands. 8

grade: in the system context, teacher's standardized evaluation of a student's work. 5, 15, 18, 23, 28, 35, 39, 40

header: section of an e-mail that consists of fields such as summary, sender, receiver, subject, and other information about it. 25, 32, 43, 44

homework: see **assignment**. 5, 13–15, 20–24, 27, 31–36, 38, 42

Integer: In the system context, data type that represents some finite subset of the mathematical integers. 42–44

Internet: worldwide, publicly accessible series of interconnected computer networks that transmit data by packet switching using the standard Internet Protocol. 6, 21, 25, 32, 36, 37

mailbox: In the system context, folder where messages are delivered and stored, e-mail equivalent of a Letter box. 36

mark: see **grade**. 5, 15, 18, 23, 28, 35, 39, 40

POP3: Post Office Protocol version 3, Internet standard protocol used to retrieve e-mail from a remote server. 6, 9, 25, 32

plain text: ordinary "unformatted" sequential file readable as textual material without format processing. 16, 36

professor: in the system context, the person in charge of collecting, commenting and grading the assignments from a course. 5–8, 13, 14, 19–23

RES system: system used in the CSC (Computer Science and Communications) department from the KTH (Kungliga Tekniska Högskolan) university for the storage and management of courses information (grades, attending students...). 5, 17, 18, 27–29, 39, 40, 44

SMTP: Simple Mail Transfer Protocol, Internet standard for e-mail transmissions across the Internet. 6, 9, 21, 37

Secure Shell: Secure Shell, network protocol that allows data to be exchanged over a secure channel between two computers. 44

String: In the system context, data type that represents an ordered sequence of characters. 42–44

student: in the system context, the person that must elaborate and send the assignments of the course to the professor. 5, 6, 13, 14, 16, 17, 19, 21, 24, 26, 31, 33, 35, 36, 38, 42

teacher: see **professor**. 5–8, 13, 14, 19–23

UTF-8: 8-bit UCS/Unicode Transformation Format, variable-length character encoding for Unicode, an industry standard allowing computers to consistently represent and manipulate text expressed in most of the world's writing systems. 16, 36

user: see **professor**. 5–8, 13, 14, 19–23

Windows XP: Windows eXPerience, line of operating systems developed by Microsoft for use on general-purpose computer systems, including home and business desktops, notebook computers, and media centers. 6, 7, 18

wxWidgets: widget toolkit for creating graphical user interfaces (GUIs) for cross-platform applications, enabling a program's GUI code to compile and run on several computer platforms with minimal or no code changes. 8

XML: eXtensible Markup Language, general-purpose markup language that provides a way to combine a text and extra information about it. 8, 9, 18, 39, 40, 42

4 User requirements definition

4.a User requirements

Definition and management of the courses

Creation of a course

1.1.1. Creation of a course

The system shall provide a mean of creating a course. The course will contain the list of students and all the necessary information about assignments (grade and some possible comments). There can exist many different courses at the same time.

Rationale: This is needed because there can exist assignments from a range of courses and students associated with different courses. This will help in organizing the courses and the homework.

Importing students list

1.2.1. Importing students list

The system shall allow the user to import a list of students from an external system into a created course.

Rationale: The amount of students attending a course can be very large, adding all of these students by hand into the system can be very time consuming. Adding a way of importing a list of students will help.

Defining an assignment

1.3.1. Defining an assignment

The system shall provide a mean of defining an assignment that will be associated with an existing course. There shall be a way to assign rules to each defined assignment. The rules that can be defined are the number of files attached in the e-mail, the format of these files and the expected e-mail subject associated with the assignment.

Rationale: Since there can be many assignments in a course there need to be a way of representing this in the system.

Collection and organization of assignments

Importing assignments

2.1.1. Input from students

The system shall allow students to submit their assignments by e-mail.

Rationale: E-mail is a widely spread and convenient way of communication. All students registered on a course should have access to a computer with e-mail capability. Most universities provide computers for the students to use at school.

2.1.2. Downloading assignments

The system shall download only those e-mails that relate to defined assignments, and respect the consistence of the used e-mail account.

Rationale: The user shall be able to use the e-mail account for other purposes. E-mails that do not relate to an assignment can be handled using his or her normal e-mail client.

2.1.3. Identifying e-mails

The system shall identify which e-mail corresponds to a defined assignment and which not.

Rationale: Identify which e-mails are actually assignments can be a very time consuming task, so this process is very convenient to be automated.

2.1.4. Manual handling of assignments

The system shall provide a way to manually import assignments from students.

Rationale: Students may submit their work without using e-mail. There might be a good reason for this so, the system must provide some flexibility.

2.1.5. Non-functional product requirement

The e-mail receiving process shall be secure, so the communications with the e-mail server shall be encrypted.

Organization of the assignments

2.2.1. Deleting assignment

The system shall provide a method to delete assignments that have been already stored in it.

Rationale: A teacher can be, for many reasons, interested in removing the assignment of a specific student (for example, the teacher can detect some problem, remove the assignment and ask the student to send it again).

2.2.2. Organizing assignments

The system shall organize downloaded assignments into a folder hierarchy.

Rationale: A simple folder hierarchy is a natural and convenient way to organize files on a computer. With this organization it can be easy, for example, to create a CD backup with all the assignments of a course.

Editing and grading of assignments

Opening files

3.1.1. Opening files

The user shall be able to open any documents fetched by the system. The user shall inform the people sending documents what file format to use.

Rationale: To be able to open any kind of documents the system should not put any constrains on specific file formats. There will be a problem if the user receives a document which the users system does not have support for, thus informing solves the problem.

Commenting files

3.2.1. Commenting files

The user shall be able to make and store comments on any documents fetched by the system.

Rationale: Commenting documents excludes the edition of the document itself. Instead, the comments will be stored by the system with the corresponding assignment.

Grading assignments

3.3.1. Grading assignments

The system shall provide a mean of grading the assignments associated with the students.

Rationale: Since homework assignments usually are graded, this needs to be represented in the system as well.

3.3.2. Non-functional product requirement

The grading format shall be general enough to accept grades in a very different formats. The system shall allow the user to enter the grades in alpha-numerical code.

Rationale: There are a lot of different grading formats in different countries, and the system should not be limited to a specific one.

Answer to the assignments

Delivery confirmation

4.1.1. OK confirmation

The system shall notify the sender of an assignment of the reception of this. The system shall automatically send an e-mail to the sender of the assignment as a delivery confirmation at the moment his assignment is received by the system.

Rationale: Delivery confirmations help the students that have sent their assignments to verify that these have really arrived to their destination, and can be used as a receipt in case there is some problem with the delivery.

4.1.2. Error notification

The system shall notify the sender of an assignment of possible errors when the e-mail does not fit the expected pattern. In this case, the system shall send automatically an e-mail to the sender of the assignment warning him that the received e-mail either does not contain the expected number of attached files or the format of some of these is not the expected one.

Rationale: This basic error notification can save a lot of work to the user of the system, that otherwise should notify the errors manually, and gives a chance to the sender to realize early of his error and fix them.

4.1.3. Non-functional product requirement

The delivery confirmation and error notification e-mails shall be encoded as plain text, using the encoding standard UTF-8.

4.1.4. Non-functional product requirement

The e-mail sending process shall be secure, so the communications with the e-mail server shall be encrypted.

Sending feedback to students

4.2.1. Sending feedback to students

The system shall provide a method to send feedback to students by e-mail.

Rationale: Students submit their assignments by e-mail, so it is reasonable to send feedback using the same way of communication.

4.2.2. Composition of feedback

The system shall automatically compose the feedback e-mails. The feedback will be composed from teachers comments and grading. The system shall only send the feedback on request from user.

Rationale: The information is already in the system. If the teacher would have to manually compose feedback e-mails it would be unnecessarily time consuming, and a possible source of errors.

Reporting

General state of a specific course

5.1.1. Reports about the general state of a specific course

The system shall create reports with information about the general status of a specific course. These reports shall include a list of students and assignments, and basic information about each assignment.

Rationale: A general status report is useful to give the user an overview of the course, the assignments that have been already submitted, etc.

State of a specific student

5.2.1. Reports about the state of a specific student

The system shall create reports with information about the status of a specific student attending a course. These reports shall include a list of all the assignments of the student, with some detailed information about each of them.

Rationale: A report on the status of a student can be useful to justify a student's grade at the end of the course, and to keep track of the individual work of each student during the course.

State of a specific assignment

5.3.1. Reports about the state of a specific assignment

The system shall create reports with information about the status of specific assignments. These reports shall include a list of all the students implied in the assignment, with some detailed information about each of them.

Rationale: The user needs this functionality to manage assignments at an individual level (keep track of the percentage of submitted assignments, etc.).

RES system link

Importing from the RES system

6.1.1. List of students import

The system shall provide a mean of creating a list with the needed data of all the students registered in a course from the RES system. The system shall obtain the data from the students that are registered in the desired course, and create a list with this information that, finally, can be used to define a course.

Rationale: In a large course, with tens or even hundreds of students, the task of introducing manually their names, e-mails, etc. can be so tedious that it could even discourage the potential users of the system. As in almost every university there is an internal system as RES where teachers can extract the list of students from, it is important to provide a way to automate this task and coordinate both systems.

6.1.2. Non-functional product requirement

The imported list of students must be stored as an XML (Extensible Markup Language) file that can be directly recognized and used in the definition of a course.

Exporting to the RES system

6.2.1. List of grades export

The system shall provide a mean of updating the grades of the students in the RES system with the information obtained from a specific course.

Rationale: It is essential to avoid the need to repeat the task of manually setting all the assignment grades twice, for the same reason as given in the user requirement 6.1.1.

6.2.2. Non-functional product requirement

The exported list of grades must be stored as an XML file.

General non-functional requirements

Product requirements

7.1.1.

The files created by the system shall be implemented using XML.

7.1.2.

The system shall compile and run, at least, in the next operative systems: Windows XP, GNU/Linux (Ubuntu 7.x distribution) and FreeBSD 6.x.

Organizational requirements

7.2.1.

A first stable version of the system shall be ready to use in April 2008.

External requirements

7.3.1.

The system shall be licensed under a GPL (General Public License).

4.b Use cases

Definition and management of the courses

1.1. Create a course

- **Primary Actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: to create a course in the system.
  - Student: to ensure a safe handling of the assignments.
- **Precondition:** None
- **Minimal Guarantee:** The system remains stable.
- **Success Guarantee:** The course is created.
- **Trigger:** The teacher tells the system to create a course.
- **Main Success Scenario:**
  1. The teacher tells the system to create a course.
  2. The teacher selects the name for the course.
  3. The system validates the course name.
  4. The system creates the course.
- **Extensions:**
  3a. The name for the course is invalid:
  3a1. The system notifies the teacher
  3a2. The system repeats from step 2 in Main Success Scenario.

1.2. Import students list

- **Primary Actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: to import the student list.
  - Student: to ensure a safe handling of the assignments.
- **Precondition:** A course have been created
- **Minimal Guarantee:** The course data remains stable.
- **Success Guarantee:** The list of students is imported.

- **Trigger:** The teacher tells the system to import a list of students.
- **Main Success Scenario:**
  1. The teacher tells the system to import a list of students.
  2. The teacher selects which course the students in the list should be added to.
  3. The teacher selects the location of the list of students.
  4. The system validates the format of the list of students.
  5. The system adds the students in the list to the selected course.
- **Extensions:**

  3a. The location of the list of students is invalid.

  3a1 The system notifies the teacher and terminates proceedings.

  4a. The format of the list of students is invalid.

  4a1. The system notifies the teacher and terminates proceedings.

1.3. Define an assignment

- **Primary Actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: to successfully define an assignment in the system
  - Student: safe handling of assignments.
- **Precondition:** A course have been created
- **Minimal Guarantee:** The course data remains stable.
- **Success Guarantee:** An assignment is defined.
- **Trigger:** The teacher tells the system to define an assignment.
- **Main Success Scenario:**
  1. The teacher tells the system to define an assignment.
  2. The teacher selects an existing course.
  3. The teacher names the assignment.
  4. The system validates the assignment name.
  5. The teacher sets up rules about the assignment.
  6. The system adds the assignment to each student in the course.
- **Extensions:**

  4a. The assignment name is invalid:

  4a1. The system notifies the teacher.

  4a2. The system repeats from step 3 in Main Success Scenario.

Collection and organization of assignments

2.1. Import an assignment manually

- **Primary Actor:** Teacher
- **Stakeholders and Interests:**
  - Teacher: to import a student assignment that has not been received by e-mail.
  - Student: to submit homework assignment.
- **Precondition:** At least one course defined with students and assignments. A selected student and a selected assignment.
- **Minimal Guarantees:** An assignment is marked.
- **Success Guarantees:** An assignment is marked and their files are registered.
- **Trigger:** Teacher tells the system to import an assignment.
- **Main Success Scenario:**
  1. Teacher tells the system to import an assignment.
  2. The teacher selects files to import.
  3. The system copies the files to the student assignment directory.
  4. The system marks the assignment.
- **Extensions:**

  2a. There are no files to import:

  2a1. The system terminates import process.

2.2. Send feedback to student by e-mail

- **Primary Actor:** Teacher
- **Stakeholders and Interests:**
  - Teacher: to send feedback on student homework assignment.
  - Student: to receive feedback on homework assignment.
- **Precondition:**

  A selected student.

  A selected assignment with comments and a grade.

  Connection to the Internet.

  E-mail account with SMTP support.

- **Minimal Guarantees:**
  An attempt to connect to the e-mail account.
  A e-mail is composed and saved for later sending.
- **Success Guarantees:** A e-mail is composed and sent to the student.
- **Trigger:** The teacher tells system to send feedback.
- **Main Success Scenario:**
  1. The teacher tells system to send feedback.
  2. The system composes an e-mail message from comments and grade.
  3. The system connects to the e-mail account.
  4. The system sends the e-mail.
- **Extensions:**
  3a. A connection to the e-mail account could not be established.
  2a1. The system saves the message for later sending and terminates process.

Edition and grading of assignments

3.1. Open a file

- **Primary actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: needs to review the assignment, which is in the file.
  - Student: to ensure a safe handling of the assignments.
- **Preconditions:** The filename must be specified, the file must exist.
- **Minimal guarantee:** The file remains uncorrupted.
- **Success guarantee:** The specified file is opened.
- **Trigger:** The teacher choose one of the documents attached in a email.
- **Main success scenario:**
  1. The teacher choose one of the documents attached in the email.
  2. The system reads the file format of the document.
  3. The system executes the appropriate application to view the file.
- **Extensions:**
  3a. The file format is not a known to the users system.
  3a1. The system notifies teacher and terminates proceedings.

3.2. Comment a file

- **Primary actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: wants to add comments to the documents.
  - Student: wants to get his or her documents reviewed.
- **Preconditions:** A selected assignment for a defined student.
- **Minimal guarantee:** The course data remains stable.
- **Success guarantee:** The document is commented.
- **Trigger:** The teacher tells system to add comments to an assignment.
- **Main success scenario:**
  1. The teacher tells system to add comments to an assignment.
  2. The teacher writes comments.
  3. The system saves the comments to the assignment.

3.3. Grade an assignment

- **Primary Actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: to grade the assignment that belong to the student in mind
  - Student: to ensure a safe handling of the assignments.
- **Precondition:** A course have been created. An assignment have been created. The student who is going to be graded have to exist in the system.
- **Minimal Guarantee:** The course data remains stable.
- **Success Guarantee:** The student's assignment is graded.
- **Trigger:** The teacher selects a course.
- **Main Success Scenario:**
  1. The teacher selects a course.
  2. The teacher selects a student.
  3. The teacher selects an assignment.
  4. The teacher inputs the grade.
  5. The system validates the grade format.
  6. The system registers the grade for the assignment.
- **Extensions:**

  5a. The grade is invalid:

  5a1 The system notifies the teacher.

  5a2. The system repeats from step 4 in Main Success Scenario.

Answer to the assignments

4.1. Send an assignment

- **Primary actor:** Student
- **Stakeholders and Interests:**
  - Student: to ensure that his assignment will be received.
  - Teacher: to ensure that all assignments are correctly built before reviewing them.
- **Preconditions:**
  - The student knows the e-mail address where he must send his assignment.
  - The student knows the expected format (number of files and type of these) of the e-mail.
- **Minimal Guarantee:** The e-mail is received and the system tries to identify it.
- **Success Guarantees:**
  - The system registers the assignment of the student.
  - The student receives a receipt confirming that the assignment has been registered.
- **Main success scenario:**

  1.The student sends his assignment (e-mail with some attached files) to the teacher's e- mail address via his usual e-mail client.

  2 The system verifies that the e-mail corresponds to some defined assignment.

  3 The system verifies that the e-mail comes from some defined student.

  4 The system verifies that the e-mail respects the expected format.

  5 The system verifies that the e-mail has been sent in time.

  6 The system stores the assignment and replies the student with a receipt confirming that the assignment has been registered.
- **Extensions:**

  2a The e-mail cannot be identified as a defined assignment delivery:

  2a1 The e-mail is ignored by the system and no reply is given to the student.

  3a The sender cannot be identified as a registered student:

  3a1 The e-mail is ignored by the system and a warning e-mail is sent to the student.

  4a The number of attached files in the e-mail or the expected format of these does not fit the expected pattern:

  4a1 The e-mail is ignored by the system and a warning e-mail is sent to the student.

  5a The e-mail has been sent after the set deadline for the assignment:

  5a1 A warning e-mail is sent to the student, the delay is registered by the system and the sequence continues.

4.2. Import assignments from e-mail

- **Primary Actor:** Teacher
- **Stakeholders and Interests:**
  - Teacher: to receive students homework assignments.
  - Student: to submit homework assignment.
- **Precondition:**

  At least one course defined with students and assignments.

  Connection to the Internet.

  E-mail account with POP3 support.
- **Minimal Guarantees:** An attempt to connect to the e-mail account.
- **Success Guarantees:** A connection to the e-mail account is established, the account is checked for unread messages, the assignments are marked and stored, e-mail confirmations are sent.
- **Trigger:** The teacher tells the system to download new assignments
- **Main Success Scenario:**

  1. The teacher tells the system to download new assignments
  2. The system connects to the e-mail account.
  3. The system checks the account for unread messages.
  4. The system validates headers for unread messages.
  5. The system downloads unread messages with valid header.
  6. The system copies downloaded messages and attachments to students assignment directories.
  7. The system marks the assignments.
  8. The system sends confirmation e-mails to students.
- **Extensions:**

  2a. A connection to the e-mail account could not be established.

  2a1. The system notifies it to the teacher and terminate import process.

  3a. No unread messages in the e-mail account

  3a.1 The system terminates import process.

  4a. Header is not valid.

  4a.1 The system marks header as non valid.

  4a.2 The system continues validating headers.

  8a A message does not have the expected files attached

  8a.1 The system appends an error report to the confirmation e-mail

Reporting

5.1. View the general state of a specific course

- **Primary actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: wants to read relevant information of an ongoing course.
- **Preconditions:** None
- **Minimal guarantee:** The course data remains stable.
- **Success guarantee:** The information of the specified course is retrieved.
- **Trigger:** The teacher tells system to show course status.
- **Main success scenario:**
  1. The teacher tells system to show course status.
  2. the teacher selects a course
  3. The system retrieves the information.
  4. The system converts the information to the right format.
  5. The system displays the information..
- **Extensions:**

  3a. The system fails to find the course in the database.

  3a1. The system notifies teacher and terminates proceedings.

5.2. View the state of a specific student

- **Primary actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: wants to read relevant information about an student.
- **Preconditions:** None
- **Minimal guarantee:** The course data remains stable.
- **Success guarantee:** The information about the specified student is retrieved.
- **Trigger:** The teacher tells system to show student status.
- **Main success scenario:**
  1. The teacher tells system to show student status.
  2. The teacher selects a student.
  3. The system retrieves the information.
  4. The system converts the information to the right format.
  5. The system displays the information.
- **Extensions:**

  3a. The system fails to find the student in the database.

  3a1. The system notifies teacher and terminates proceedings.

5.3. View the state of a specific assignment

- **Primary actor:** Teacher
- **Stakeholders and interests:**
  - Teacher: wants to read relevant information about an assignment.
- **Preconditions:** None
- **Minimal guarantee:** The course data remains stable.
- **Success guarantee:** The information about the specified assignment is retrieved.
- **Trigger:** The teacher tells system to show assignment status.
- **Main success scenario:**
  1. The teacher tells system to show assignment status.
  2. The teacher selects assignment.
  3. The system retrieves the information.
  4. The system converts the information to the right format.
  5. The system displays the information.
- **Extensions:**

  3a. The system fails to find the assignment in the database.

  3a1. The system notifies teacher and terminates proceedings.

RES system link

6.1. Import list of students of a course from the RES system

- **Primary actor:** Teacher
- **Stakeholders and Interests:**
  - Teacher: to get the updated list of students for his course, so he does not need to introduce it manually in the system.
  - RES system managers: to ensure that the information stored in the RES system remains consistent after the operation.
- **Preconditions:**

  – The course have been previously defined in the RES system and the students are registered there.

  – The teacher has authorization to access to the RES system.
- **Minimal guarantee:** The information in the RES system remains stable.

- **Success Guarantee:**

  – The teacher gets a file with a list of information about students that have registered n the RES system for the desired course.

- **Main success scenario:**

  1. The teacher starts the application that communicates with the RES system and provides the needed information about the course whose students list he wants to get.

  2 The system gets the information from the RES system and creates a file with the list of students that can be later imported by System.

- **Extensions:**

  1a The RES system does not contain the required course:

  1a1 The system warns the teacher and asks for the information about the course again.

  1a2 The teacher supplies the needed information.

6.2. Export grades of a course to the RES system

- **Primary actor:** Teacher

- **Stakeholders and Interests:**

  - Teacher: to copy the grades from System to the RES system, so he does not need to duplicate them manually.
  - RES system managers: to ensure that the information stored in the RES System remains consistent after the operation.

- **Preconditions:**

  – The course have been previously defined in the RES system and the students are registered there.

  – The graded assignments have been previously defined in the RES system.

  – The teacher has authorization to access to the RES system.

- **Minimal guarantee:** The information in the RES system remains stable.

- **Success Guarantees:**

  – The system stores the grades information, exported from the system to a file, in the RES system.

- **Main success scenario:**

  1 The teacher starts the system and opens a created course.

  2 The teacher tells the system to export the grades of the course to a file and specifies the file name and path.

  3 The system creates the file with all the necessary information in order to be able to store it in the RES system.

  4 The teacher starts the application that communicates with the RES system, providing the grades file to it.

  5 The system stores all the information about grades that the file contained in the RES system.

- **Extensions:**

  2a The system detects that there are not assigned grades for the opened course:

  2a1 The system warns the teacher and aborts the export.

  4a The RES system does not contain the course specified in the given file:

  4a1 The system warns the teacher and aborts the process.

5 System architecture

5.a Control-flow model

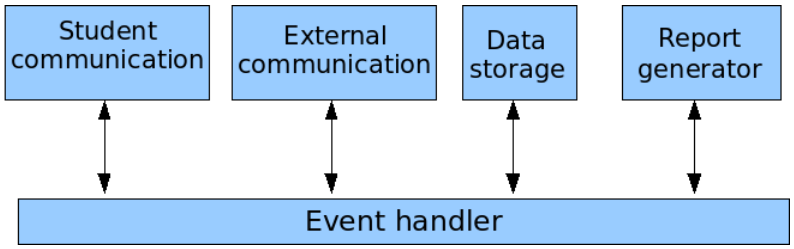

Figure 2: Control-flow model of the system

5.b Data-flow model

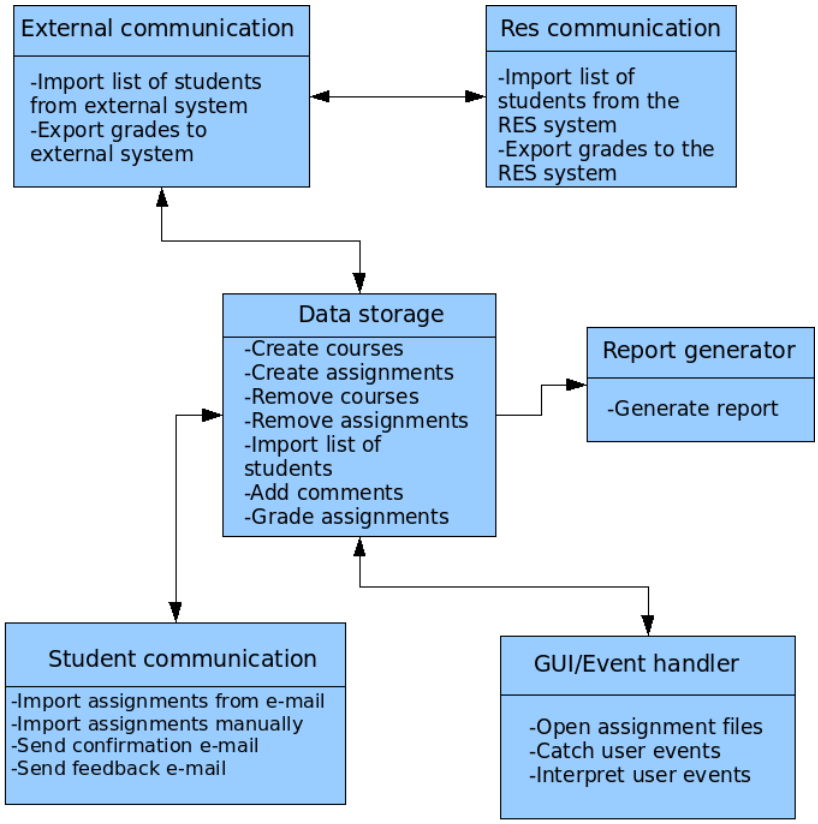

Figure 3: Data-flow model of the system

6 System requirements specification

Definition and management of the courses

1.1. Creation of a course

- **Function:** Creation of a course.
- **Description:** Create a course in the system.
- **Inputs:** A course name.
- **Source:** Course name is an input from the user.
- **Outputs:** A created course in the system.
- **Action:** If the course name is valid a course is created from the input name.
- **Requires:** Nothing.
- **Pre-condition:** None.
- **Side-effects:** None.
- **Satisfied user requirements:** 1.1.1

1.2. Importing students list

- **Function:** Importing students list.
- **Description:** Import a student list from the local computer.
- **Inputs:** Location of the student list. A course.
- **Source:** Location of the student list is an input from the user. The course is selected by the user.
- **Outputs:** Name and e-mail of the students.
- **Action:** If the location of the list of students is valid and the format of the list is valid the students are loaded into the selected course.
- **Requires:** The course in mind have to exist.
- **Pre-condition:** None.
- **Side-effects:** None.
- **Satisfied user requirements:** 1.2.1

1.3. Defining an assignment

- **Function:** Defining an assignment.
- **Description:** Define an assignment in the system and the possibility to set up rules for the assignment.
- **Inputs:** An assignment name. A course. Expected subject of the e-mails. Number of files and file format.
- **Source:** All inputs from the user.
- **Outputs:** A defined assignment.
- **Action:** Define an assignment in the input course for all the students in the course. The assignment is associated with the input rules (number of files and file format) and the expected subject line in the e-mail.
- **Requires:** The course in mind have to exist.
- **Pre-condition:** None.
- **Side-effects:** None.
- **Satisfied user requirements:** 1.3.1

Collection and organization of assignments

2.1. Importing assignments

2.1.1. From the e-mail account

- **Function:** Import an assignment by e-mail.
- **Description:** Imports an assignment sent to the e-mail account of the teacher.
- **Inputs:** An e-mail.
- **Source:** E-mail server.
- **Outputs:** Assignment extracted from the e-mail.
- **Destination:** Organize e-mails.
- **Action:** Connect to e-mail server. Download header from server. Validate e-mail. If email is valid download e-mail and call Organize e-mails.
- **Requires:** Connection to the Internet. E-mail account with POP3 support.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 2.1.1, 2.1.2, 2.1.3

2.1.2. Manually

- **Function:** Manual importation of an assignment.
- **Description:** Import assignments manually.
- **Inputs:** Course, student, assignment and possibly some files.
- **Source:** User input.
- **Outputs:** None.
- **Action:** Mark assignment and copy files to the folder hierarchy. The folder is determined from course, student and assignment.
- **Requires:** A defined course with students and assignments.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 2.1.4

2.2. Organization of the assignment

2.2.1. Organization of the assignment

- **Function:** Deleting an assignment.
- **Description:** Deletes files and messages accidentally associated with an assignment.
- **Inputs:** Assignment.
- **Source:** User input.
- **Outputs:** None.
- **Action:** Delete files and messages associated with an assignment.
- **Requires:** A defined course with students and marked assignments.
- **Pre-condition:** None.
- **Post-condition:** The selected files and messages are deleted.
- **Side effects:** None.
- **Satisfied user requirements:** 2.2.1

2.2.2 Organization of the assignment

- **Function:** Organize assignments.
- **Description:** Saves files and messages from the assignment in a folder hierarchy.
- **Inputs:** A received assignment.
- **Source:** Import assignments by e-mail.
- **Outputs:** None.
- **Action:** Save the message and attached files in folder hierarchy. The folder is determined from course, student and assignment. Mark the assignment.
- **Requires:** A defined course with students and assignments.
- **Pre-condition:** None.
- **Post-condition:** Files are saved to disk.
- **Side effects:** None.
- **Satisfied user requirements:** 2.2.2

Edition and grading of assignments

3.1. Opening files

- **Function:** Opening files.
- **Description:** Opens the file attached in the given email.
- **Inputs:** The filename for the attached file.
- **Source:** E-mail client.
- **Outputs:** None.
- **Destination:** None.
- **Action:** The appropriate application to be used to open the file is launched.
- **Requires:** That the users system has support to view the file.
- **Pre-condition:** The user must have informed the email senders of which file format to use.
- **Post-condition:** None.
- **Side effects:** File format constrains are dependent on the users system.
- **Satisfied user requirements:** 3.1.1

3.2. Commenting files

- **Function:** Commenting files.
- **Description:** The system gives the user the possibility to write comments for the opened file.
- **Inputs:** Users comment.
- **Source:** The user.
- **Outputs:** None.
- **Action:** The user writes a comment about the file. The file is saved and closed.
- **Requires:** the relevant information.
- **Pre-condition:** A file has been opened.
- **Post-condition:** The written comment is saved in the storage system with relevant information of who the comment was intended to, to which file and assignment.
- **Side effects:** Only general comments suits best.
- **Satisfied user requirements:** 3.2.1

3.3. Grading assignments

- **Function:** Grading assignments.
- **Description:** To grade a assignment.
- **Inputs:** A grade. A course, a student in the course and a assignment from the student.
- **Source:** All are inputs from the user.
- **Outputs:** A graded assignment.
- **Action:** Grade the selected assignment (that belongs to a student, who belongs to a course) with the input grade.
- **Requires:** The course in mind have to exist in the system, the student have to exist in the system and the assignment have to exist in the system.
- **Pre-condition:** None.
- **Side-effects:** None.
- **Satisfied user requirements:** 3.3.1, 3.3.2

Answer to the assignments

4.1. Delivery confirmation

- **Function:** Assignment delivery validation.

- **Description:** Validates a given e-mail as a correct assignment delivery, and notifies the sender either of the correct validation and reception of this or of the errors that it has.
- **Inputs:** An e-mail e.
- **Source:** The mailbox of the e-mail account associated with the user.
- **Outputs:** Validity of e as a correct assignment delivery.
- **Action:** e is validated following the steps reflected in the figure 4.

	Validation	**If true**	**If false**
1	e corresponds to a defined assignment	Next step	Ignore e, return false
2	e comes from a defined student	Next step	Ignore e, reply a warning, return false
3	e respects the expected format	Next step	Ignore e, reply a warning, return false
4	e has been sent in time	reply a receipt, return true	Reply a receipt with a warning, return true

Figure 4: Validation steps of an e-mail

The test that the function applies to e in the step 3 is to check if the number of attached files and the format of these corresponds to what is expected to be delivered in the assignment (pattern previously defined by the user).

- **Requires:** Internet connection, in order to automatically reply to e if necessary.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** A receipt e-mail or a warning e-mail can be replied to the sender of e.
- **Satisfied user requirements:** 4.1.1, 4.1.2, 4.1.3, 4.1.4
- **Special requirements:** The receipt and warning e-mails shall be encoded as plain text, using the encoding standard UTF-8.

4.2. Sending feedback

4.2.1. Send feedback to student

- **Function:** Send feedback to student.
- **Description:** Sends feedback on an assignment to the student.
- **Inputs:** Assignment, student, comments and grades.
- **Source:** User input.
- **Outputs:** An e-mail.
- **Destination:** e-mail server.

- **Action:** Compose feedback e-mail and send to student.
- **Requires:** Connection to the Internet. E-mail account with SMTP support.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 4.2.1

4.2.2. Composition of feedback

- **Function:** Composition of feedback.
- **Description:** Puts together a feedback e-mail from comments and grade.
- **Inputs:** Comments, grade.
- **Source:** User input.
- **Outputs:** An e-mail.
- **Destination:** Send feedback to an student.
- **Action:** Insert comments and grade into predefined template.
- **Requires:** None.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 4.2.2

Reporting

5.1. General state of a course

- **Function:** Show the general state of a specific course.
- **Description:** Provides an overview of the course, assignments and students basic statistical data to the user.
- **Inputs:** A course code or name.
- **Source:** Retrieves relevant information from the storage system.
- **Outputs:** An overview of the specified course.

- **Destination:** None.
- **Action:** Either the course exist in the storage system and information can be retrieved or the user is prompted with a "failed to identify course" message.
- **Requires:** A course identifier.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 5.1.1

5.2. State of a specific student

- **Function:** Show the state of a specific student.
- **Description:** Retrieves information about a specific student.
- **Inputs:** A student code or name.
- **Source:** Retrieves relevant information from the storage system.
- **Outputs:** Information about the specified student.
- **Action:** Either the student exist in the storage system and information can be retrieved or the user is prompted with a "failed to identify student" message.
- **Requires:** A student identifier.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 5.2.1

5.3. State of a specific assignment

- **Function:** Show the state of a specific assignment.
- **Description:** Retrieves information about the specified assignment.
- **Inputs:** An assignment code or name.
- **Source:** Retrieves relevant information from storage system.
- **Outputs:** Information about the specified assignment.
- **Action:** Either the assignment exist in the storage system and information can be retrieved or the user is prompted with a "failed to identify assignment" message.

- **Requires:** An assignment identifier.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 5.3.1

RES system link

6.1. Importing list of students from the RES system

- **Function:** Get list of students from the RES system.
- **Description:** Gets the list of students and associated information of a specific course from the RES system, storing this in a new file that the system is able to recognize.
- **Inputs:** A course identifier.
- **Source:** The course identifier is an input from the user.
- **Outputs:** A list of students (names, identifiers and associated information).
- **Destination:** Independent application that interacts with the RES system.
- **Action:** If there exists a course defined with the given identifier, a list of students that have registered in the RES system for the specified course is obtained.
- **Requires:** Authorization to access to the RES system.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 6.1.1
- **Special requirements:** The list of students must be stored as an XML (Extensible Markup Language) file that can be directly recognized and used in the definition of a course.

6.2. Export grades from the system

- **Function:** Export grades from the system.
- **Description:** Exports the grades stored for an specific assignment and course into a list of grades that can be used later to introduce them in the RES system.
- **Inputs:** An existing course, an existing assignment.

- **Source:** All inputs from the user.
- **Outputs:** A list of grades for a specific assignment (names of students and grades obtained).
  Action: A list with the grades of all the students in the given assignment is created.
- **Requires:** Nothing.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 6.2.1
- **Special requirements:** The exported list of grades must be stored as an XML file that the application that interacts with the RES system can recognize.

6.3. Store grades in the RES system

- **Function:** Store grades in the RES system.
- **Description:** Stores the given grades of a list of students in a specific assignment.
- **Inputs:** A list with the grades of all the students in a specific assignment.
- **Source:** A file provided by the user.
- **Outputs:** The RES system is updated with the given information.
- **Destination:** Independent application that interacts with the RES system.
- **Action:** If the given list is correctly formatted and the given course, assignment and students are defined in the RES system, the given grades of a list of students in a specific assignment are stored in it.
- **Requires:** Authorization to access to the RES system.
- **Pre-condition:** None.
- **Post-condition:** None.
- **Side effects:** None.
- **Satisfied user requirements:** 6.2.2
- **Special requirements:** The input (list of grades) must be stored as an XML file.

7 System evolution

These are the fundamental assumptions on which the system is based:

- Simplification of organizing, managing and keeping track of course related information such as assignments.
- The user needs to have a valid and working email address and required data to be able to receive and send e-mails.
- Information exchange is made through Internet using e-mail.
- Configurable e-mail filter system thus making AAMS to only receive relevant emails.

Anticipated changes will be probably due to:

- **Hardware evolution:** should only affect AAMS in a positive way such as increased speed and reliability.
- **Changing user requirements:** university teachers might want to comment directly into the document.
- **Changing system requirements:** by using custom made plug-in modules. AAMS should be able to interact with various universities internal assignment or homework grading system.

8 Appendices

8.a Logical organization of data

The data that is stored by the system will be organized in three different XML files: "data.xml", "settings.xml" and "unsentemails.xml".

data.xml

This file contains four sections: CourseList, FileInfoList, StudentInfoList and AssignmentInfoList.

- **CourseList section:** This section contains an item (name Course) for each course. Each item is a section itself. The attribute for this item is name.
  - **The name attribute (String):** Name of the course.
  - **Course section:** This section contain an item (named Student) for each student in the course. Each item is a section itself. The attribute for this item is id.
  - **The Id attribute (Integer):** Each student item has an id number corresponding to an item in the StudentInfoList section with the same id number.
  - **Student section:** This section contain an item (named Assignment) for each assignment for the student. Each item is a section itself. The attributes for this item are id and comment.
  - **The id attribute (Integer):** Each assignment item has an id number corresponding to an item in the AssignmentInfoList section with the same id number.
  - **The comment attribute (String):** A comment on the assignment.
  - **Assignment section:** This section contains an item (named File) for each assignment for the student. The attributes for this item are id, filename and comment.
  - **The id attribute (Integer):** Each file item has an id number corresponding to an item in the FileInfoList section with the same id number.
  - **The filename attribute (String):** Name of the file.
  - **The comment attribute (String):** A comment on the file in the assignment.
- **FileInfoList section:** This section contains items that hold the information for each file associated with an assignment. The attributes for each item are id and extension.
  - **The id attribute (Integer):** Each item in this section has a unique integer identifying the file information.
  - **The extension attribute (String):** The extension of the file.
- **StudentInfoList section:** This section contains items that hold the information for each student in a course. The attributes for each item are id, e-mail and name.
  - **The id attribute (Integer):** Each item in this section have a unique integer identifying the the student information.
  - **The email attribute (String):** The e-mail of the student.
  - **The name attribute (String):** The name of the student.
- **AssignmentInfoList section:** This section contain items that hold the information for each assignment. The attributes for each item is id, name and emailheader.

- **The id attribute (Integer):** Each item in this section has a unique integer identifying the assignment information.
- **The name attribute (String):** The name of the assignment.
- **The emailheader attribute (String):** The header that is used to identify the e-emails sent by the student.

The figure 5 contains an example of the data.xml file.

```
<CourseList>
  <Course name="Advanced computer science">
        <Student id="1">
          <Assignment id="1" comment="">
            <File id="1" filename="" comment="" />
            <File id="2" filename="" comment="Good work!." />
          </Assignment>
          <Assignment id="2" comment="">
            <File id="3" filename="" comment="" />
          </Assignment>
        </Student>
        <Student id="2">
          <Assignment id="1" comment="">
            <File id="1" filename="" comment="" />
            <File id="2" filename="" comment="" />
          </Assignment>
          <Assignment id="2" comment="">
            <File id="3" filename="" comment="Section 3 needs some more work." />
          </Assignment>
        </Student>
  </Course>
  <Course name="Advanced computer science 2">
  </Course>

</CourseList>

<FileInfoList>
  <File id="1" extension="pdf" />
  <File id="2" extension="pdf" />
  <File id="3" extension="txt" />
</FileInfoList>

<StudentInfoList>
  <Student id="1" email="kalle@kth.se" name="Kalle Karlsson" />
  <Student id="2" email="david@kth.se" name="David Davidsson" />
</StudentList>

<AssignmentInfoList>
  <Assignment id="1" name="Homework 1" emailheader="Homework 1" />
  <Assignment id="2" name="Homework 2" emailheader="Homework 2" />
</AssignmentInfoList>
```

Figure 5: Example of the data.xml file

settings.xml

This file contains the data for the different settings used in the system. It contains two items, Pop3 and Smpt. Each of these items has three attributes: address, port, login and password.

- **The address attribute (String):** Address of the server.
- **The port attribute (Integer):** Port number of the server.
- **The login attribute (String):** Login name for the e-mail account.
- **The password attribute (String):** Password for the e-mail account.

The figure 6 contains an example of the settings.xml file.

```
<Settings>
  <Pop3 adress="pop3.kth.se" port="110" login="kalle" password="ytghs54jh" />
  <Smpt adress="smpt.kth.se" port="25" login="kalle" password="ytghs54jh" />
</Settings>
```

Figure 6: Example of the settings.xml file

unsentemails.xml

This file contains the data for the unsent e-emails. It contains an item for each of the unsent e-mails. The attributes for each item are destination, header and content.

- **The destination attribute (String):** Contains the e-mail address of the receiver.
- **The header attribute (String):** Contains the header for the e-mail.
- **The content attribute (String):** Contains the text or content of the e-mail.

The figure 7 contains an example of the unsentemails.xml file.

```
<UnsentEmails>
  <Email destination="kalle@kth.se" header="Re: Homework 1" content="Most looks ok, but..." />
  <Email destination="lisa@kth.se" header="Re: Assignment 4" content="Good work!" />
  <Email destination="david@kth.se" header="Re: Report" content="This arrived late, you..." />
</UnsentEmails>
```

Figure 7: Example of the unsentemails.xml file

8.b Structure and interface of the RES system

The communication with the RES system implies two basic operations over it: to read the course information from it, in order to get a list of students, and to write grades corresponding with assignments in it. The main objective is to make the user able to perform both operations from the same point where AAMS is working.

The usual way to access RES system is through several terminal commands that must be executed in some specific machines from the department (those from the networks *my*, *faun* and *cyklop*). These commands can be executed locally or remotely, for example through a Secure Shell (SSH). The RES commands have this syntax:

res [<options> ...] <operation> <course> [<additional arguments> ...]

Typical operations are: *checkin*, *view*, *edit*, ... The basic operations needed for our system could be, in principle, *print* (to get the list of students) and *edit* (to edit the course information and introduce the grades of the assignments).

9 Index

List of Figures

Index