Cmpt 275

Simon Fraser University
School of Computing Science

Cmpt 275 Project

Date: 2015-05-31

Team name: Eleven ®

Project Deliverable #: 2
Project Deliverable Name: Requirements, Draft 1
Phase(s) covered by this deliverable: Release 1 Requirements Specification

---

Phase leader(s): Susan Hamilton
Team members and Student #:

Yinglun Qiao	301191776
Janice Sargent	301160485
Zhi Cheng	301184486
Susan Hamilton	301209641
Seong Jun Kim	301154246
Nari Shin	301178863
Fan Liu	301168674
Te Lun Chen	301200832
Roy Chan	301202770

Grade:

Revision History

Revision	Status	Publication/ Revision Date	By:
1.0	Created a skeleton for the document	*Saturday* May 23, 2015	Fan Liu Yinglun Qiao
1.1	Added 1-1.1.1.1, 1-1.1.2, 1-1.1,3, 1-1.2 and created 1-2, 1-3, 1-4, 1-5, 1-6	*Thursday May 28, 2015*	Te Lun Chen Zhi Cheng
1.2	Added 1-2.3, 1-2.3.1, 1-3.1.1, 1-3.1.2	*Friday* May 29, 2015	Janice Sargent
1.3	Edited 1-4 and 1-5 and added 1-4.1 to 1-4.3, 1-5.1 to 1-5.8 and removed 1-3.2, 1-3.4 , 1-5.6.3	*Saturday* May 30, 2015	Nari Shin
1.4	Edited and added to 1-1.1.1.1.1, 1-2.1.1, 1-2.1.2, 1-3,1-5, 1-7, 2-1.2, 2-1.4, 2-5	*Saturday* May 30, 2015	Susan Hamilton
1.5	Modified non-function requirements, and moved most of 1-7.1 of functional requirements into the non-functional group	*Saturday* May 30, 2015	Yinglun Qiao, Fan Liu, Janice Sargent, Seong Jun Kim
1.6	Added and Modified List 1, List 2, List 3, List 4	*Saturday* May 30, 2015	Roy
1.7	Edited functional list 1-4, added 5.7-5.9, 7.4 7.5 7.7 7.8 7.9, removed 3.3 (already in 5.4)	*Sunday* May 31, 2015	Nari Shin
1.8	Edited general format, spelling, and grammar Edited and added to 1-2.1.2, 1-2.1.4, 1-3.1.2, 3.2 (removed 3.2.2 as already present),1-5, 1-7.3.1,removed 2-2.2.1.5, and section 4	*Sunday* May 31, 2015	Susan Hamilton

Table of Contents

Part I

Requirements Specification

1 - Functional Requirements

1. Manage Account Information

1.1. A System Administrator manages all accounts for this system. This includes any or all of the following actions:

1.1.1. Creating an account

1.1.1.1. For each account, the following attributes need to be provided:

1.1.1.1.1. Role(s)

1.1.1.1.1.1. Administrator

1.1.1.1.1.2. System Administrator

1.1.1.1.1.3. Administrative Assistant

1.1.1.1.1.4. Instructor

1.1.1.1.1.5. Teacher Assistant

1.1.1.1.2. Name

1.1.1.1.3. Employee ID

1.1.1.1.4. User ID

1.1.1.1.5. Temporary password

1.1.2. Modifying an account

1.1.3. Deleting an account

2. Manage Courses

2.1. An Administrative Assistant manages all courses for this system. This includes any or all of the following actions:

2.1.1. Creating a course

2.1.1.1. For each course, the following attributes need to be provided:

2.1.1.1.1. Course number

2.1.1.1.2. Course name

2.1.1.1.3. Start date

2.1.1.1.4. End date

2.1.1.2. For each course, the following may be added additionally:

2.1.1.2.1. Instructor name

2.1.1.2.2. Teaching Assistant names

2.1.1.2.2.1. Can have 0 or more

2.1.1.2.3. Employee ID for markers

2.1.1.2.4. List of students' names

2.1.1.2.5. Computer ID of students

2.1.1.2.6. Student number

2.1.2. Modifying a course

2.1.2.1.1. Can only modify the items listed in 2.1.1

2.1.3. Deleting a course

2.1.3.1. Cannot be deleted once activities has been added

2.1.4. Copying a course

2.1.4.1. Only the attributes of the courses listed under 2.1.1.1 and 2.1.1.2 can be copied excluding:

2.1.4.1.1. List of students

2.1.4.1.2. List of students' computer ID

2.1.4.1.3. List of students' student number

2.1.4.1.4. Start date

2.1.4.1.5. End date

2.1.4.1.6. Any materials added by Instructor

2.2. An Administrative Assistant cannot modify any activities or grades for any courses

2.3. An Administrative Assistant can assign Instructors, students and Teaching Assistants to courses

2.3.1. The system will use students' computer ID and their student number to identify students

3. Manage Activities

3.1. An Instructor can manage the activities to be graded during each of his/her courses. This includes any of the following actions:

3.1.1. Creating an activity

3.1.1.1. The following attributes must be specified:

3.1.1.1.1. Activity Name

3.1.1.1.2. Activity Language: English, French, C++, Java etc.

3.1.1.1.3. Activity Type: essays, multiple choice, problem sets, programming

3.1.1.1.4. Activity Solution

3.1.1.1.4.1. File path is provided by Instructor

3.1.1.2. Additional specifications that may be added and modified later

3.1.1.2.1. A rubric

3.1.1.2.2. Description

3.1.1.2.3. Due date

3.1.1.2.4. Input file

3.1.1.2.5. Programming tests

3.1.1.2.6. Late policy

3.1.1.2.7. Bonus marks

3.1.2. Modifying an activity

3.1.2.1. May modify information specified under 3.1.1

3.1.3. Deleting an activity

3.1.4. The system will update the detailed information of each of the activities entered by the Instructor

3.1.5. The system will release information to the Instructor only relating to his/her course

3.2. Copying activities

3.2.1. Some or all activities can be copied from the Instructor's previous courses or activities from the same course taught by a different Instructor

4. Manage Rubric

4.1. An Instructor manages all rubrics on this system. This includes all of the following actions:

4.1.1. Creating a rubric

4.1.1.1.1. Rubric is created after activity has been created

4.1.2. Modifying a rubric

4.1.3. Copying a rubric

4.1.3.1.1. Copy from any previous offerings of the same course

4.2. A rubric must have:

4.2.1. Path to its stored directory

4.2.2. Description of items to be graded

4.2.3. Weights for each item

4.3. At any time the Instructor can set or update the rubric

5. Manage Solution and Grades

5.1. Markers are Teaching Assistants and Instructors who have the following actions:

5.1.1. Can view the work of each student only one at a time

5.1.2. Can add comments directly to submitted work

5.1.2.1. Except for multiple choice activities

5.1.3. Can enter grades for each item in rubric after viewing student's work related to that portion of the rubric

5.1.4. Are able to grade or regrade any student's work

5.1.4.1. There is no time limit on how long a marker is able to regrade an assignment

5.2. Grading programming assignments

5.2.1. The system accepts a series of tests to be executed for testing submissions' correctness

5.2.1.1. Each test includes:

5.2.1.1.1. Input files and console inputs

5.2.1.1.2. Correct output files and console output

5.2.2. The system helps an Instructor assemble input and output files

5.2.3. The system compares correct output and output of student's code

5.2.4. The system should display related parts of rubric to marker

5.2.4.1. A marker will enter a grade for that portion

5.2.5. The system must be able to compile in C, C++, JAVA, and PYTHON

5.2.5.1. The system must be compatible with the environments such as Visual Studio and Eclipse

5.2.5.2. If a code does not compile, the system must notify the marker

5.3. Grading multiple choice tests

5.3.1. An Instructor provides answer keys and a list of weights

5.3.2. An Instructor can modify answer keys and weights if necessary

5.4. Grading essays and problem sets

5.4.1. A marker can manually look for a section of a student's work and the reference section from the solution file, and compare it to the corresponding section of the rubric

5.4.1.1. The marker can then give the corresponding mark for this item in the rubric

5.5. Plagiarism

5.5.1. Dealt with in the Course Management System

5.6. The system must be able to handle bonuses and deductions

5.6.1. Early Bonuses

5.6.1.1. There are up to three possible early submission dates. Each date is specified by the number of hours early, and the percentage of the grade to be added

5.6.2. Late Policy

5.6.2.1. There are up to three possible late submission dates. Each date is specified by the number of hours late, and the percentage of the grade to be deducted

6. Manage Transferring of Documents

6.1. The system receives students' submissions from the Course Management System. This includes the following:

6.1.1. A list of students or groups who submit the assignments

6.1.2. The time of submission generated by the Course Management System

6.2. The csv files generated by the system can be loaded back into the Course Management System

6.2.1. These files can be generated whenever an instructor requests them

6.3. Commented work on calculations or derivation problems, and essays can be returned to students through the Course Management System

7. Access the System

7.1. An administrator can view and update all material added by other users

7.1.1. Material refers to all data in the system except account information

7.2. The system checks the authentication of each user for the login to the system

7.3. The system requires the following identification items to log in a user:

7.3.1. User ID

7.3.2. Password

7.4. The system provides new temporary password for users who fail to log in

7.4.1. The processes listed from 2.2.1.2.1 - 2.2.1.2.3 are carried out

7.5. The system is able to support multiple users

7.6. Instructors are only able to see the list of courses they are assigned to

7.6.1. When a course is selected, they are able to see a list of existing activities, and they can manage or mark those activities

7.7. Logs of the system

7.7.1. Logs contain a list of any changes to the system

7.8. Notifications relating to a course

7.8.1. Instructor may send out notifications regarding courses

7.9. Installation and Backup

7.9.1. Managed by System Administrator

7.9.2. Including backups of

7.9.2.1. Course information

7.9.2.2. Rubric

7.9.2.3. Activities

2 - Non-Functional Requirements

I - Quality Requirements

1. **Usability**
   1.1. The system provides an interface to help the Instructors to assemble all necessary input and output files for programming assignments
      1.1.1. There is no limit on the number of programming tests
      Allows a marker to enter the path to the solution code, sample
      1.1.2. inputs, sample outputs, console input, and console outputs
   1.2. If a user has multiple roles, when they have logged onto the system they will have an option to select which role they wish to log on as. Once the user has selected a role, then the user will only log on to the role that the user has chosen.

2. **Dependability**
   2.1. Reliability
      2.1.1. The system keeps a backup of activity solutions in case of the solution link provided by the instructor changes
   2.2. Security
      2.2.1. User Logins
         2.2.1.1. User ID
            2.2.1.1.1. If the user has multiple roles, then the user needs to choose her role to sign in
         2.2.1.2. Password
            2.2.1.2.1. A System Administrator sends by courier a temporary password for each new account
            2.2.1.2.2. The user is prompted upon initial login to change to a new password on their own
            2.2.1.2.3. If the user forgets the password, or fails to login after 6 unsuccessful attempts, the above steps 2.2.1.2.1 and 2.2.1.2.2 are repeated
         2.2.1.3. The employee IDs that are used to uniquely identify each employee should be a 12-digit number
         2.2.1.4. The user IDs that are used to uniquely identify each user in the system cannot exceed 8 characters

3. **Performance**
   3.1. The system grades only one assignment at a time
   3.2. Switching between a student's submitted work, the Instructor's solution, and the activity's rubric should be no more than one click away

4. **Supportability**
    4.1. Maintainability and Portability
        4.1.1. Database Management
            4.1.1.1. Backups occur regularly throughout the term, and formal backups occur at the end of each term
            4.1.1.2. Persistent data will be stored in a MS SQL database. All data will be kept for 5 years
    4.2. Adaptability
        4.2.1. Ability to add any directories as a marker wishes.
            4.2.1.1. Eg, create directories to store solutions.

II - Constraints

1. **Implementation Requirements**
    1.1. The group have agreed to implement the system by using Java in Eclipse as the main programming language
    1.2. The system should run on a Windows platform

2. **Interface Requirements**
    2.1. Format of csv file
        2.1.1. The csv file generated by the system contains one student in each row, and one item in the rubric in each column
        2.1.2. For multiple choice tests, each row in the csv will be results for one student. Each row will provide the choices made by the student for each of the questions on the multiple choice test.

3. **Operations Requirements**
    3.1. Students’ codes must be able to compile when marking
        3.1.1. Versions of the programming assignment environments should be the same as those in CSIL
    3.2. Student cannot be graded by multiple markers simultaneously. If two markers are trying to grade the same student at the same time, one of the them will be given a warning and blocked
    3.3. If a rubric is modified during marking then all students' activities must be remarked
    3.4. Solution files to activities within a course is provided by instructors via a path

4. **Packaging Requirements**

4.1. Installation will occur initially by System Administrator to set up the system

5. Legal Requirements

5.1. The system development must comply with SFU regulations

3 - Prioritization of Functional Requirements

3.1 **LIST 1** (core or critical)

- **Account Management (1)**
    - Account Creation (1.1.1)
    - Account Deletion (1.1.3)
    - Attributes
        - Roles (Administrator, System Admin etc.) (1.1.1.1.1)
        - Name, Employee ID, User ID, Password (1.1.1.1.2 - 1.1.1.1.5)

- **Course Management (2)**
    - Course Creation (2.1.1)
        - Number, Name, Start/End dates (2.1.1.1.1-2.1.1.1.4)
        - Administrative Assistant can assign Instructors, students and Teaching Assistants (2.3)
- **Activity Management (3)**
    - Activity Creation (3.1.1)
        - After course creation
            - Name, Language, Type and Solution (3.1.1.1.1-3.1.1.1.4)
                - Information and Files provided by Instructor (3.1.1.1.4.1)
- **Rubric (4)**
    - Rubric Creation (4.1.1)
    - Directory (4.2.1)
    - Description and weights for items to be graded (4.2.2)
- **Grading (5)**
    - Markers can view student submissions one student at a time (5.1.1)
    - Markers can enter grades for each item in rubric after viewing student submission (5.1.3)
    - Programming Assignments (5.2)
        - Input files and console inputs (5.2.1.1.1)
        - Correct output files and console output (5.2.1.1.2)
        - Help Instructor assemble I/O files (5.2.2)
        - Compare I/O of student's code submissions (5.2.3)
    - Multiple Choice (5.3)
        - Answer Key, Weights (5.3.1)
- **System (7)**
    - Login/Logout (7.2)
    - Identification (7.3)
        - Computer ID and student number (7.3.1)

3.2 LIST 2 (most important of the remaining)

- **Account Management (1)**
  - Account Modification(s) (1.1.2)
- **Course Management (2)**
  - Course Modification(s) (2.1.2)
  - Attributes that can be added after course has been created
    - Instructor name (2.1.2.1.1)
    - Employee ID (2.1.2.1.2)
    - Teaching Assistant names (2.1.2.1.3)
    - Teaching Assistant employee ID (2.1.2.1.4)
    - List of students' names (2.1.2.1.5)
    - Computer ID of students (2.1.2.1.6)
    - Student number (2.1.2.1.7)
  - Course Deletion (2.1.3)
- **Activity Management (3)**
  - Activity Modifications (3.1.2)
  - Activity Deletion (3.1.3)
  - Attributes that can be added or modified after activity has been created (3.1.1.2)
    - Rubric(3.1.1.2.1)
    - Description (3.1.1.2.2)
    - Due date (3.1.1.2.3)
    - Input file (3.1.1.2.4)
    - Programming tests (3.1.1.2.5)
- **Rubric (4)**
  - Modify rubric (4.1.2)
- **Grading (5)**
  - Multiple Choice (5.3)
    - Modify Answer keys and Weights (5.3.2)
- **System (7)**
  - Supports multiple users (7.5)

3.3 LIST 3 (needed but less important to be functioning of core features of the system)

- **Activity Management (3)**
  - copying from previous courses or activities from the same course (3.2.1)
- **Rubric (4)**
  - copying rubric from previous course offerings (4.1.3)
- **Grading (5)**
  - Add comments to student's work (5.1.2)

    - Ability to regrade student’s work (5.1.4)
    - Late Policy (5.6.2)
        - Up to three late submission dates (5.6.2.1)
- **Documents (6)**
    - Can be loaded back into course management system (6.2)
        - CSV generated when Instructor needs it (6.2.1)
- **System (7)**
    - Login Failure (7.4)
        - System Administrator sends temporary password (7.4.1)

3.4 **LIST 4** (not necessary to the successful and efficient operation of the core system)

- **Grading (5)**
    - Plagiarism (5.7)
        - Instructor may use outside tools (5.7.1)
    - Bonus Marks (5.6)
        - Early bonuses (5.6.1)
            - Up to three possible early submission dates (5.6.1.1)
- **System (7)**
    - Logs (7.7)
        - containing any changes to system (7.7.1)
    - Notifications (7.8)
    - Installation and backup (7.9)
        - Managed by System Administrator (7.9.1)
        - Including backups of (7.9.2)
            - Course information (7.9.2.1)
            - Rubric (7.9.2.2)
            - Activities (7.9.2.3)

4 - Prioritization of Non-Functional Requirements

x.x : Quality Requirements   x.x : Constraints

4.1 **LIST 1** (core or critical)

- **Dependability (2)**
    - Security (2.2)
        - User Logins (2.2.1)
            - User ID and password are required (2.2.1.1)
- **Operations Requirements** (3)
    - Students' code can be compiled (3.1)
        - Same versions of programming environment as CSIL (3.1.1)
    - Student cannot be graded simultaneously (3.2)
- **Packaging Requirement** (4)
    - The System Administrator installs and sets up system (4.1)
- **Legal Requirements** (5)
    - The system is subject to SFU regulations (5.1)

4.2 **LIST 2** (most important of the remaining)

- **Performance (3)**
    - One assignment is graded at a time (3.1)
- **Dependability (2)**
    - Security (2.2)
        - User Logins (2.2.1)
            - Employee IDs are uniquely identify by a 12-digit number (2.2.1.2)
- **Supportability (4)**
    - Adaptability (4.2)
        - Marker can add directories to store information (4.2.1)
- **Implementation Requirement (1)**
    - The system runs on Windows platform (1.2)
- **Operations Requirements (3)**
    - Activities are remarked if rubric is changed during marking (3.3)
    - Solution files are provided via a path (3.4)
- **Interface Requirements (2)**
    - Csv file (2.1)
        - Csv file has students in rows and activities in columns (2.1.1)
        - Csv file of multiple choice test has students in rows and choices in columns (2.1.2)

4.3 **LIST 3** (needed but less important to be functioning of core features of the system)

- **Usability (1)**
  - Input and output files assemble interface for programming assignments is provided (1.1)
    - No limit on the number of programming assignments (1.1.1)
    - Marker can enter the path to the solution code, sample inputs, sample outputs, console input, and console outputs (1.1.2)
- **Dependability (2)**
  - Reliability (2.1)
    - The system keeps back up of activity solution (2.1.1)
  - Security (2.2)
    - User Logins (2.2.1)
      - User ID is no more than 8 characters (2.2.1.3)
- **Supportability (4)**
  - Maintainability (4.1)
    - Database management (4.1.1)
      - Backup (4.1.1.1)
      - Data are stored in MS SQL database and kept for 5 years (4.1.1.2)

4.4 **LIST 4** (not necessary to the successful and efficient operation of the core system aka only the client wants this)

- **Usability (1)**
  - If user has multiple roles, the user needs to select one when log in (1.2)
- **Dependability (2)**
  - Security (2.2)
    - User Logins (2.2.1)
      - Allow maximum 6 unsuccessful login attempts (2.2.1.4)
- **Performance (3)**
  - Switch between tasks in one click away (3.2)
- **implementation requirement (1)**
  - The system uses Java as programming language (1.1)

Part II

Requirements Analysis / System Models

Requirements Analysis / System Models

Part III

Appendix 1

Agenda - Deliverable 2: Release 1 Requirements Specification

Date: May 28, 2015
Time: 3:30 PM
Location: Outside CSIL lab (we might discuss outside, so you can bring laptops)

Preparations:

- Attend SVN meeting on Thursday May. 28, 2015 at 1:30pm.

- Take a look at our Software Requirements Specification Document: Release 1
- Try to fill in if you have any ideas or share any suggestions on skype
- Bring ideas regarding functional requirements, non-functional requirements, and prioritization of functional requirements to the meeting.

Objectives:

- This meeting mainly focuses on brainstorming functional requirements, non-functional requirements, and prioritization of functional requirements together as a team.
    - These requirements are specified on the course website: http://www.cs.sfu.ca/CourseCentral/275/jregan/assignments/PD2Req1.html
- We work together to identify the requirements that are clear, and ones that are unclear.
    - For the unclear requirements or any other type of questions, we discuss them and can group them together, and send team members to discuss those questions with the client later if needed.
- Each member should try to make some contributions to the document.

Status report:

- Has document for Deliverable 2 created?
- Has Deliverable 2 been received by everyone?
- Has everyone taken a look at Deliverable 2 document?
- If applicable, has meeting minutes on May.20 been check into SVN?

Discussion Items: – 40 minutes

- Requirements discussion - 30 minutes
    - Brainstorm ideas on requirements
- Assign duties for requirements. - 10 minutes
    - Each member is assigned or a specific section in functional requirement, non-functional requirement, and prioritization of functional requirement.

Wrap Up: Assign new action items for the next meeting.

Meeting Minutes1 - Deliverable 2: Release 1 Requirements Specification

Date: May 28th, 2015
Time: 15:45 - 16:40
Duration: 55 minutes
Location: SFU Burnaby Campus CSIL Lab

Attendance:

Te Lun Chen
Yinglun Qiao
Susan Hamilton
Fan Liu
Zhi Cheng
Nari Shin
Roy Chan
Seong Jun Kim
Janice Sargent

Absent: None

Items Covered:

1. Review of Agenda
2. Review of Software Requirements Specification Document: Release 1
3. Modifications of all requirements according to Client Requirements Gathering Meeting
4. List of the modified requirements
5. Discussions of which requirements are functional requirements
6. Discussions of which requirements are non-functional requirements
    - § For item 5 and 6, our group got confused with the definition of functional requirements and non-functional requirements. We have made a decision writing an email to the professor to ask for help. We have decided to have a group meeting through Skype on Sat. May 30 at 17:30 to further determine which requirements are functional and which are non-functional requirements plus the prioritizations of functional requirements.
7. Dividing our group into 3 subgroups and assigning each group a task:
   Functional Requirement: Oscar, Andy, Nari, Susan
   Non-Functional Requirement: Janice, Colin, Josh
   Prioritization of functional requirement: Roy, Fan