Department of Computer Science and Software Engineering
Concordia University

COMP 354 — Introduction to Software Engineering
Winter 2020
Term Team Project Description
Kakuro

1 Objective

The objective of this project is to apply software engineering best practices to develop a desktop application in an iterative and time-boxed manner, that plays the Kakuro game and also allows user to play. With each iteration, a new increment of the Kakuro System must be specified, modeled, built, tested and presented.

The focus of the project is on learning the software process and the inter-relatedness of the activities in a project. The process emphasizes:

- test-driven development, to ensure good understanding of requirements, and working software;
- agile development, to encourage iteration, to divide the work into small chunks, and to allow students to improve their skills from one iteration to the next;
- object-oriented software development, in particular software systems as collections of collaborating objects, with a focus on responsibility-driven design, an emphasis of separating interfaces from implementations, and the use of patterns; and
- communication, by being team-based, and requiring basic documentation using UML for domain model, use cases, architecture and design, and testing.

2 Project Description and Expected Features

Your team has been hired to develop the Kakuro desktop game based on the popular puzzle game called Kakuro.

Wikipedia has an article on Kakuro `https://en.wikipedia.org/wiki/Kakuro` and a collection of 10-by-10 Kakuro games can be found at `http://www.menneske.no/kakuro/10x10/eng/`.

The desktop application in Java will have a basic graphical user interface (GUI) and be built using the Model-View-Controler (MVC) architecture.

This is a prototype application for the client. The prototype should limit its scope and not try to be ultra-intelligent in how it plays the game.

As the developers, the client wishes you to decide on a set of use cases, and to categorize them as *“must have”*, *“good to have”*, or *“optional”*.

For your team work in this course, you should aim to have at least one use case per team member. However, for the first increment, you should decide on three basic use cases that can be implemented **in one week**.

Constraints — Application:

- Standalone JAVA application
- JAVA SWING GUI (or similar)
- SQLite DB (or text files) for data storage

Constraints — Work Environment:

- Java as the programming language;
- Eclipse as the integrated development environment (IDE);
- JUnit for unit testing in Java;
- Rational Rose for UML modeling.
- git software repository for team on github
- version control
- Latex for documentation

3 Team Formation and Organization

This is an important activity in the course — doing a team project. The most important aspect of the project is experiencing the process of software development in order to appreciate how the various phases interact, and to see the roles of the deliverables. The second most important aspect is to experience the group dynamics and interpersonal relationships of team work.

The project is to be undertaken in teams of about 9–12 (randomly assigned) members and consists of building a challengingly large Java application. Each team has three roles: organizer, coder, and documenter. Each role is fulfilled by 2-3 students. If the team has more than 9 students then there is an additional role: quality control. The project is divided into three iterations or increments. After each iteration, the resulting application will be demonstrated in the lab. Team members are required to rotate roles after each iteration. Each student should assume responsibility for one use case.

Each team will maintain a software repository at github with version control. All software will be maintained at the repository under version control. All documents including minutes of meetings and individual student diaries will be maintained at the repository under version control.

In addition, each student is required to keep a diary of their individual project activities and to submit their diary at each stage of the project.

Roles

Organiser: is responsible for coordinating activities across the whole project, but specifically for the current increment; and for maintaining good communication amongst team members; mainly through maintaining the software repository. The team should have a clear workplan and schedule with each member's tasks and deadlines clearly specified.

Coder: is responsible for developing test cases and unit tests for the code that they develop, as we follow TDD (test driven development); and for developing the code for their increment, so that it passes all tests.

Each student throughout the project is responsible for one use case that they have implemented. This includes maintaining the unit tests, the code, and ensuring accurate documentation of the use case, its design, and its tests in the three submitted documents.

Documenter: is responsible for developing the documentation for the increment; including diagrams; and ensuring the documentation complies with the project as a whole. Note that the content for the document is the joint responsibility of the team. Documents will be done using Latex.

Quality Control: is responsible for reviewing and improving aspects of the project. This includes developing additional test cases, improving existing test cases; reviewing source code, refactoring source code where appropriate; reviewing and improving documents for the current increment, and ensuring all documents are maintained to be consistent with the project as a whole.

Each subteam assumes the role of either developer, tester or documenter for the duration of an iteration. Team members rotate their roles such that each member will play each of three roles.

It is important to understand that the project is a team effort and that it may be required to "loan" members from one subteam to another in order to meet the deadline. Note that there exist strong dependencies between the subteams.

Every team is responsible for establishing "ground rules" in order to minimize conflicts. In order to ensure on-time delivery, the internal delivery schedule should allow sufficient "slack" for mistakes and re-work. It is recommended to assign tasks as early as possible, so individuals can schedule their other work.

4 Project Plan and Deliverables

The Kakuro System is to be developed in three iterations. Each iteration will produce a running system. Starting with a set of core features, with each iteration additional features will be added and/or the implementation of existing features will be modified. Each iteration will take three to four weeks. Each iteration will have a document as a deliverable: a requirements document, a design document, and a test plan and report respectively for iteration one, two, and three. Each iteration will have source code as a deliverable including unit tests.

Team members will rotate roles across the three iterations. An iteration should involve one new use case for each Coder for that iteration. That Coder will be responsible for that use case.

Submission dates for the three iterations are in week 5, week 9, and week 12 of semester. Exact deadlines for submission to the EAS (Electronic Assignment Submission) system are listed on the course web page.

Your team will be required to do a preliminary presentation of your system and document in the lab one week prior to submission. There will be a formal presentation of your submission in the lab following the submission deadline.

Iteration 1

Document: Software requirements specification (for all user stories). Each team will be responsible to independently refine the user stories into software requirements. (Your team is your customer.)

Tutorial Week 2: It is important to review **domain modeling** and **use cases** in the tutorial with your team; sketch a draft domain model for the project; and decide on the three use cases for iteration 1.

Laboratory Week 2: It is **important** to have a team meeting during this lab: (1) to get to know each other, your semester schedule, work commitments etc; (2) to discuss and agree to roles for iteration 1; (3) to assign tasks and deadlines for individual team members, especially the Coder assignment to the selected use cases for iteration 1; (4) plan how Organisers will set up git repository; get everyone's contact info for communication; plan minutes and diaries; (5) discuss the timeline between now and the presentation of iteration 1 in the Laboratory of Week 4.

Tutorial Week 3: Discuss the Model-View-Controler (MVC) architecture for the project.

Laboratory Week 3: Documenters should review Latex and create a sample document. Coders should review JUnit and create sample unit tests. The team should meet and resolve any misunderstandings for Iteration 1 about the domain model and use cases; hear how the Coders plan to implement their assigned use case; hear what issues the Documenters have, and what content/information they need. Everyone can help create sample data as needed by the unit tests. Review the timeline overall for iteration 1, assigned tasks and deadlines.

Tutorial Week 4: Discuss TDD (test-driven development); how test cases are related to use cases and scenarios; how test cases (unit tests) drive implementation;

Laboratory Week 4: Present your working version of the system for iteration 1. The team meeting should review what needs to be done over the next week to get the application ready for submission; the team should review status of the document, and review the timeline overall for iteration 1, assigned tasks and deadlines, for both the software and the document to be ready for submission.

Tutorial Week 5: Laboratory Week 5: Last minute scrum to resolve remaining work for software and document submission. Do not forget to write minutes of meetings and individual diaries, and place these on the repository under version control. Do not forget to submit software, document, diary. Submit to EAS under required category: Documents should be submitted as "project 1"; Source code as "programming_assignment 1"; diary as "theory_assignment 1".

Submission at end of Week 5.

Personal Study Week 5 and 6: Review course material for Quiz 1. Pay particular attention to how the lectures relate to your project work. Web page should have questions to guide your study. Review what you covered in tutorials for Iteration 1: domain model, use cases, TDD, MVC.

Iteration 2

Document: Architectural and detailed design specification (for all user stories)

Tutorial Week 6: Review course material for Quiz 1 or Catch your breath (depending on when your tutorial is scheduled).

Laboratory Week 6: Team meeting to celebrate submission of Iteration 1; discuss rotation of roles for Iteration 2; select use cases for Iteration 2; and agree on task assignments, deadlines, and overall timeline. Possible de-brief on what worked well and what did not work well on Iteration 1. Possible plan to fix any remaining issues in software for Iteration 1.

Tutorial Week 7: Discuss how to document software architecture and design; Especially your architecture and design.

Laboratory Week 7: Documenters should review Latex and create a sample document. Coders should review JUnit and create sample unit tests. The team should meet and resolve any misunderstandings for Iteration 2 about the domain model and use cases; hear how the Coders plan to implement their assigned use case; hear what issues the Documenters have, and what content/information they need. Everyone can help create sample data as needed by the unit tests. Review the timeline overall for iteration 2, assigned tasks and deadlines.

Tutorial Week 8: Discuss RDD (responsibility driven design); discuss design patterns; discuss architectural patterns.

Laboratory Week 8: Present your working version of the system for iteration 2. The team meeting should review what needs to be done over the next week to get the application ready for submission; the team should review status of the document, and review the timeline overall for iteration 2, assigned tasks and deadlines, for both the software and the document to be ready for submission.

Tutorial Week 9: Laboratory Week 9: Last minute scrum to resolve remaining work for software and document submission. Do not forget to write minutes of meetings and individual diaries, and place these on the repository under version control. Do not forget to submit software, document, diary. Submit to EAS under required category: Documents should be submitted as "project 2"; Source code as "programming_assignment 2"; diary as "theory_assignment 2".

Submission at end of Week 9.

Iteration 3

Document: Test plan and test report

Tutorial Week 10: Review material on testing; validation and verification; contents of test report.

Laboratory Week 10: Team meeting to celebrate submission of Iteration 2; discuss rotation of roles for Iteration 3; select use cases for Iteration 3; and agree on task assignments, deadlines, and overall timeline. Possible de-brief on what worked well and what did not work well on Iteration 2. Possible plan to fix any remaining issues in software for Iteration 2.

Personal Study Week 10 and 11: Review course material for Quiz 2. Pay particular attention to how the lectures relate to your project work. Web page should have questions to guide your study. Review what you covered in tutorials.

Tutorial Week 11: Review how to perform system testing, integration testing, robustness testing using JUnit (and the Facade design pattern).

Laboratory Week 11: Present your working version of the system for iteration 3. The team meeting should review what needs to be done over the next week to get the application ready for submission; the team should review status of the document, and review the timeline overall for iteration 3, assigned tasks and deadlines, for both the software and the document to be ready for submission.

Tutorial Week 12: Laboratory Week 12: Last minute scrum to resolve remaining work for software and document submission. Do not forget to write minutes of meetings and individual diaries, and place these on the repository under version control. Do not forget to submit software, document, diary. Submit to EAS under required category: Documents should be submitted as "project 3"; Source code as "programming_assignment 3"; diary as "theory_assignment 3".

Submission at end of Week 12.

Tutorial Week 13: Laboratory Week 13: Final project presentations and demos will be scheduled.