Information System Implementation

420-MP6-AS

Winter 2021

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 https://www.menneske.no/kakuro/10x10/

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. For your teamwork in this course, you should aim to have at least one use case per team member.

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.
- Visual Paradigm UML modeling.

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 3-5 members and consists of building a Java application. Each team has three roles: organizer, coder, and documenter. Each role is fulfilled by 2-3 students. The project is divided into three iterations or increments. After each iteration, the resulting application will be demonstrated. Team members are required to rotate roles after each iteration. Each student should assume responsibility for one use case.

Roles

Organizer: is responsible for coordinating activities across the whole project, but specifically for the current increment; and for maintaining good communication amongst team members. The team should have a clear work plan 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. Students are responsible for 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 or doc.

Each sub-team 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 sub-team to another in order to meet the deadline. Note that there exist strong dependencies between the sub-teams.

Every team is responsible for establishing "ground rules" in order to minimize conflicts. 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.

Your team will be required to do a preliminary presentation of your system and document one week prior to submission.

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.)

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.

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

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

Week 5: Last minute scrum to resolve remaining work for software and document submission.

Iteration 2

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

Week 6: Review course material or Catch your breath.

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

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

Week 9: Last minute scrum to resolve remaining work for software and document submission.

Iteration 3

Document: Test plan and test report

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

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

Week 12: Last minute scrum to resolve remaining work for software and document submission.

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