AY 2023/2024

Politecnico di Milano

RASD: Requirement Analysis and Specification Document

Tommaso Capacci Gabriele Ginestroni

Professor
Elisabetta Di Nitto

Version 1.2
November 13, 2023

Contents

1 Introduction

1.1 Purpose

Traditional classroom teaching focuses heavily on theory and concepts, without providing sufficient opportunities for students to gain hands-on coding experience. Additionally, the student-instructor relationship is usually limited to formal lectures and exams, missing some kind of continuous evaluation, which is essential to stimulate learning.

CodeKataBattle aims to fill these gaps by facilitating competitive programming challenges that motivate students to put concepts into practice. The aim of the product is to provide an alternative to evaluate students with respect to their coding skills in a more enjoyable way: this is achieved by allowing educators to create tournaments in which students can partecipate and compete. Through integration with static analysis tools, the automated scoring system provides reliable evaluation so students can measure their improvements. By collaborating in teams, they also learn key skills like communication and source control. Indeed, partecipating to battles, students will gain experience in using GitHub as source control system.

These kind of evaluation also teaches how the test-first approach can be useful when it comes to developing a new piece of software.

For instructors, CodeKataBattle enables closer mentorship driven by qualitative code reviews, by including optional manual evaluation. In this way, educators will have the opportunity to enhance their code review skills while increasing the engagement with their students. Overall, the platform creates a virtuous cycle of learning powered by practice, feedback and community.

1.1.1 Goals

The main objectives of our system are the following:

- **G1: Allow students to participate to programming challenges in teams**
  This is the main goal of the system. Students will be able to form teams to partecipate to Code Kata Battles and collaborate through GitHub.

- **G2: Allow educators to create tournaments**
  Only educators are allowed to create tournaments, which are composed by a non predefined number of battles. Whenever a new tournament

is created, all the students of the platform are notified and can partecipate to it. After the creation of a tournament, the educator can add collaborators that will help him in the management of the tournament.

- **G3: Allow educators to create programming challenges**
  The creator of the tournament (or invited collaborators) can create programming challenges (Code Kata Battles) within the context of a tournament.

- **G4: Let the system to automatically analyze and rank submissions of students**
  Students submissions are automatically analyzed and ranked by the system, combining functional aspects, timeliness and quality level of sources. Functional aspects are measured in terms of number of unit test cases passed out of all test cases. Timeliness instead is measured in terms of elapsed time passed since the start of the battle. Finally, quality level of sources is measured in terms of code quality, whose aspects can be selected by the educator at battle creation time, and is computed by integrating with static analysis tools.

- **G5: Allow educators to manually inspect and review submissions of students**
  Educators will have the possibility to enable manual evaluation of students' submissions for a battle. If the option is enabled, the educator who created the battle, at the end of the submission phase of a battle, can review the code of all the teams and assign a score. In this case, the system will combine the automatically computed score with the manual one to produce the final battle rank.

- **G6: Allow students to track their performance**
  Teams will we able to see their current rank evolving during the battle, updated for each new submission. Additionally, at the end of each battle, the platform updates a personal tournament score of each student, that is the sum of all battle scores received in that tournament. In this way, the student can track his performance during the tournament as well.

1.2 Scope

The Code Kata Battle (CKB) platform aims at providing a collaborative environment for students to enhance their software development skills through structured practice sessions called "code kata battles". CKB facilitates an engaging learning experience by enabling students to partecipate in friendly competition while refining their programming skills.

The system should allow 2 types of accesses, one for **educators** and the other for normal **students**: educators will have the possibility to create **tournaments** and **battles**, while students will have the possibility to subscribe to tournaments and provide solutions for the battles that compose them.

Students can decide whether to participate to a battle by creating a new team or joining one, in respect with the constraints decided by the educator at battle creation time. Teams can be created by students in the context of a battle so that other students subscribed to the same tournament can freely decide to join and team up to produce a solution.

The system will facilitate the collaboration between team members by integrating with **GitHub** through its *GitHub Actions*. In this way, students will be able to collaborate on the same project and share their code.

Once the students start developing their solutions, the platform will also start evaluating them. Evaluation is performed in 2 ways:

- Mandatory **automated evaluation**; it includes:
  - functional aspects measured in terms of number of test cases that are correctly solved; unit test cases are provided by the educator when uploading the project files related to the battle
  - timeliness, measured in terms of time passed between the start of the battle and the last commit
  - quality level of the sources, extracted through **external static analysis tools** that consider multiple aspects selected by the educator at battle creation time (e.g. security, reliability, and maintainability)
- Optional **manual evaluation**: personal score assigned by the educator, who checks and evaluates the work done by students

Lastly, once the deadline for the submission of solutions expires (and after the manual evaluation has been performed in case it was expected), the system assigns to every team that participated in the battle an integer score between 0 and 100. This will concur both to the team's battle rank and to the personal score of each user in the context of the tournament the battle belongs to. At any point in time every user subscribed to CKB can see the list of ongoing tournaments and the rank of enrolled users. For a smoother experience, students will receive email notifications about the most important events, such as the availability of a new tournament and battle, or the publication of the final rank of a battle.

1.2.1 Phenomena

According to the paper "The World and the Machine" by M.Jackson and P.Zave, we can identify the application domains. The following table describes the world, shared and the machine phenomena, including the reference to which part controls the phenomena.

Phenomenon	Controlled	Shared
Teachers test their students	W	N
Evaluator decides to challenge his students	W	N
Student wants to improve his coding skills	W	N
Students may communicate with each other	W	N
Student invites other student to join his team	W	N
Student/educator subscribes to the platform	W	Y
Student/educator logins	W	Y
Educator creates a tournament	W	Y
Educator adds other educator as tournament collaborator	W	Y
Educator creates a battle within a tournament	W	Y
Educator closes the battle's consolidation stage	W	Y
Educator terminates a tournament	W	Y
Educator submits manual optional evaluation	W	Y
Student creates a team	W	Y
Student joins a team	W	Y
Student checks list of ongoing tournaments and ranks	W	Y
Student forks the GitHub repository of the code kata	W	Y
Student commits a new submission on GitHub	W	Y
Student checks the updated score of a battle	W	Y
System closes tournament's subscription phase	M	Y
System closes battle's team formation phase	M	Y
System closes battle's submission phase	M	Y
Platform creates GitHub repository and sends the link to the students of the teams subscribed to the battle	M	Y
Platform sends notification to users	M	Y
System computes and updates battle score of a team	M	Y
System computes and updates student's tournament personal score	M	Y
System evaluates quality level of a submission through static analysis	M	Y
System checks login data	M	N
System pulls new submission from GitHub	M	N
System evaluates functional aspects of a submission	M	N
System evaluates timeliness of a submission	M	N

Table 1: phenomena table

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **User:** anyone that has registered to the platform
- **Student:** the first kind of users and, basically, the people this product is designed for. Their objective is to submit solutions to battles
- **Team:** students can decide to group up and form a team to partecipate to a battle. The score assigned to the submission of a team will be assigned also to each one of its members
- **Team leader:** whoever creates a team for a battle. He is in charge of forking the GitHub repository of the battle and adding his teammates as GitHub collaborators. He also has to properly set-up the GitHub Actions workflow that will be used to automatically evaluate the submissions of the team
- **Educator:** the second kind of users. They create tournaments, set-up battles, and eventually, evaluate the solutions that teams of students have submitted during the challenge
- **Tournament:** collection of coding exercises (battles) about specific topics of a subject. Interested students can subscribe to it and participate to its battles as soon as they are published
- **Code Kata Battle (or battle):** the atomic unit of a tournament. Usually students are asked to implement an algorithm or to develope a simple project that solves the task. Each battle belongs to a specific tournament: students submitting solutions for a battle will obtain a score that will be used to compute both the team's rank for the battle and the members' tournament rank
- **Tournament collaborator:** other educator that is added by the tournament creator to help him in the management of the tournament. He can create battles and evaluate their submissions
- **GitHub collaborator:** person who is granted the access to a GitHub repository with write permission

- **Test cases:** each battle is associated with a set of test cases, which are input-output value pairs that describe the correct behavior of the ideal solution
- **Static analysis:** is the analysis of programs performed without executing them, usually achieved by applying formal methods directly to the source code. In the context of the platform this kind of analysis is used to extract additional information about the level of security, reliability and maintainability of a battle submission
- **Functional analysis:** measures the correctness of a solution in terms of passed test cases
- **Timeliness:** measures the time passed between the start of the battle and the last commit of the submission
- **Score:** to each solution is assigned a score which is computed taking into account timeliness, functional and static analysis and, eventually, manual score assigned by the educator that created the challenge. The score is a natural number between 0 and 100 (the higher the better)
- **Rank:** at the end of each battle, the platform updates the personal tournament score of each student. Specifically, the score is computed as the sum of all the battles scores received in that tournament. This overall score is used to fill out a ranking of all the students participating to the tournament which is accessible by any time and by any user subscribed to the platform
- **Notification:** it's an email alert that is sent to users to inform them that a certain event occurred such as the creation of a new tournament and battle or the publication of the final rank of a battle

1.3.2 Acronyms

- **CKB:** Code Kata Battles
- **API:** Application Programming Interface
- **UML:** Unified Modeling Language

1.3.3 Abbreviations

- **Gn:** Goal number “n”
- **Dn:** Domain Assumption number “n”
- **Rn:** Requirement number “n”

1.4 Revision History

- December 22, 2023: version 1.0 (first release)

1.5 Reference Documents

- Specification document: "Assignment RDD AY 2023-2024"
- RASD reference template: “02g.CreatingRASD”
- Paper: "Jackson and Zave: the world and the machine"
- UML official specification https://www.omg.org/spec/UML/
- Alloy official documentation: https://alloytools.org/documentation.html
- GitHub API official documentation https://docs.github.com/en/rest/guides/getting-started-with-the-rest-api?apiVersion=2022-11-28

1.6 Document Structure

- **Section 1: Introduction**
  This section offers a brief overview of the product and its domain.
  It also contains the list of definitions, acronyms and abbreviations that could be found in this document.
  Finally, there are the list of definitions, acronyms and abbreviations that could be found in this document.
- **Section 2: Overall Description**
  The second chapter offers a more detailed description of the product domain through state machine diagrams and UML class diagram. It contains also the hardware and the software costraints of the system. Finally, there is an overview of all the product features and the actors

they are built for. Some scenarios are included to better understand the expected functionalities.

- **Section 3: Specific Requirements**
  This section contains a description of functional requirement through some use cases and activity diagrams.

- **Section 4: Formal Analysis through Alloy**
  The fourth chapter is a formal analysis of the model, made through the Alloy, including a graphic representation of it obtained from Alloy Tool.

- **Section 5: Effort spent**
  The fifth and last chapter contains the time spent by each contributor of this document.

2 Overall Description

3 Specific Requirements

Organize this section according to the rules defined in the project description.

4 Formal Analysis Using Alloy

Organize this section according to the rules defined in the project description.

5 Effort Spent

Provide here information about how much effort each group member spent in working at this document. We would appreciate details here.

References