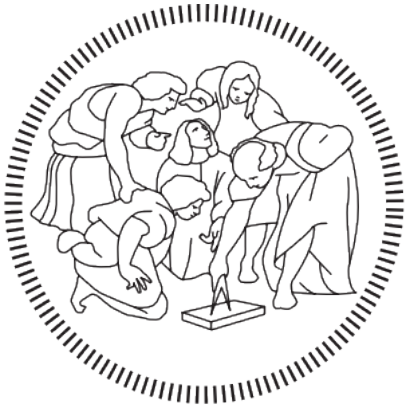

POLITECNICO
MILANO 1863

CodeKataBattle

Requirements Analysis and Specifications Document

Software Engineering 2 project
Academic year 2023 - 2024

20 October 2023
Version 0.0

Autors:
Tommaso Pasini
Elia Pontiggia
Michelangelo Stasi

Professor:
Matteo Camilli

Revision History

Date	Revision	Notes
23/10/2023	v.0.0	Document creation
	v.1.0	First relese

Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of CodeKataBattle (CKB). It provides functional and non-functional requirements for the development of the system, including use cases, features, user interaction and system constraints. This document is addressed to the developers who have to implement the requirements and could be used as an agreement between the customer and the contractors.

CodeKataBattle (CKB) is a new platform that helps students improve their software development skills by training with peers on code kata. Educators use the platform to challenge students by creating code kata battles in which teams of students can compete against each other, thus proving (and improving) their skills and soft-skills : in fact working in a team is useful to learn soft-skills such as communication and coordination.

A code kata battle is essentially a programming exercise in a programming language of choice (e.g., Java, Python). The exercise includes a brief textual description and a software project with build automation scripts (e.g., a Gradle project in case of Java sources) that contains a set of test cases that the program must pass, but without the program implementation. Students are asked to complete the project with their code. In particular, groups of students participating in a battle are expected to follow a test-first approach and develop a solution that passes the required tests. Groups deliver their solution to the platform (by the end of the battle). At the end of the battle, the platform assigns scores to groups to create a competition rank.

1.1.1 Goals

G1 Allows Students to improve their software development skills by partecipating in coding tournaments where they write programs

G2 Allows Educators to challenge Students by create coding tournaments and battles.

Gi Allows Educators to track Students' knowledge about software development, evaluating the code written during battles and the overall score obtained in tournaments

Gi Allows Educators to customize battles defining specific rules, achivements that Students can obtain and evaluation modality.

Gi Allows Students to improve their soft skills, as communication, collaboration and time management, by creating teams for coding battles

Non son sicuro siano goal che Code kata vuole risolvere

G_i Allows S to learn sw develoment divertendosi e confrontandosi con i compagni in sane competizioni con ranking

---

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

1.2 Scope

In another RASD here there are a few words

1.2.1 World Phenomena

WP1 The students fork the GitHub repository of the code kata

WP1 Students set up an automatic workflow in the GitHub repository

WP1 The students work on the code kata battle

WP1 The students push their work to the GitHub repository

WP1 An educator upload correct test cases and automation scripts

WP1 An educator evaluates the work done by the team at the end of the code kata battle

---

1.2.2 Shared Phenomena

SP_i An educator creates a tournament

SP_i An educator grants other colleagues permission to create battles

SP_i An educator creates a battle within a tournament

SP_i The students are notified of a new tournament

SP_i The students are notified of a new upcoming battle within a tournament they are subscribed to

SP_i Students use the platform to form teams

SP_i Student invite other students to join its team respecting the boundaries imposed

SP_i Student join a battle on his own

SP_i Student join a battle towards an invite -¿ Possono scegliere di essere soli oppure devono per forza

SP_i The platform sends the link of the GitHub repository to all students who are members of subscribed teams

SP_i Students are asked to fork the GitHub repository and set up an automated workflow

SP_i The forked repository's workflow notifies the platform of a new push

SP_i The platform updates the battle score of a team

SP_i The educator uses the platform to go through the sources produced by each team

SP_i The educator assigns additional score to the teams after the evaluation

SP_i The platform notifies the teams when the final battle rank becomes available

SP_i The platform updates the personal tournament score of each student

SP_i An educator closes a tournament

SP_i The platform notifies all students involved in the tournament about its end

SP_i All users see the list of ongoing tournaments as well as the corresponding tournament rank

SP_i An educator defines the gamification badges

SP_i Student visualize gained badges on its profile

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Abbreviations

G_i = i-th goal
WP_i = i-th World Phenomena
SP_i = i-th Shared Phenomena

1.3.2 Acronyms

E = educator
S = students

1.4 Revision history

1.5 Reference Documents

Assign document for A.Y. 2023-2024 "Requirement Engineering and Design Project: goal, schedule, and rules"

1.6 Document Structure

The structure of this RASD document follows six main sections:

1. **Introduction:** provides an overview of the problem at hand, the purpose of the document and the project, the scope of the domain, and introduces the main goals of the system as a solution.

2. **Overall Description:** gives a general description of the system, going into more details about its main functions. The description is assisted with the help of UML diagrams, such as class, activity and state diagram. The domain assumptions of the examined world are then explained along with any dependencies and constraints.

3. **Specific Requirements:**

4. **Formal Analysis Using Alloy:**

5. **Effort Spent:** keeps track of the time spent to complete this document. The first table defines the amount of hours used by the whole team to get important decisions and to make reviews, the other tables contains the individual effort spent by each team member.

6. **References:** lists all the documents used and that were helpfull in drafting the RASD.

2. Overall Description

2.1 Product Perspective

2.1.1 Scenarios

Creating a tournament

Chip, a professor of Algorithm and Data Structures at Mouseton Institute of Technology, prepared to teach the chapter on strings, launching the "Strings Operations" coding tournament on CKB. To expand participation, he allowed his colleague Dale to create challenges for his software engineering class. Students across classes would compete in string manipulation tasks, ranging from basic concatenation to advanced text analysis, fostering collaboration and learning. To make the tournament more interesting, Chip decided to award badges to the best performing students, so he created a badge for the student who participated in the most tournaments, one for the student who won the most battles and one for the student that wrote the most lines of code. All students already subscribed to CKB were notified of the new tournament, and they could join it from the tournament page.

Creating a battle

In order to get the students familiar with the CKB platform and its features, Chip created an easy battle for his students to practice on, called "wordcheck", that basically requires the student to implement the game wordle. He decides that the battle will last 2 weeks and that the students will be able to work in teams of 2 or 3 people; they will be able to join the battle until the last day of the battle. In addiction, he wants to give extra points to the code cleanup, so he will have to revise the code of each team at the end of the battle and assign extra points to the teams that wrote clean code. He sets all this information in the battle creation form and then he creates the battle.

Joining a battle

Huey and Dewey, two students of Chip's class, are notified of a new battle and decide to join it. Since the the more the merrier, they decide to invite their friend Louie to join them in the battle. After the registration deadline, they are notified that the battle

is about to start and they are given the link to the GitHub repository of the battle to fork and set up the automated workflow in order to be able to link their GitHub account to the CKB platform. After the automated workflow is set up, they are ready to start working on the battle.

Improving the score and obtaining a badge

Donald is a warrior of the "wordcheck" battle and he is working on the battle alone. After a fitst commit, he logs in to the CKB to check his score. He sees that he is in the 3rd position and that he is 10 points behind the leader group, that is composed by Huey, Dewey and Louie. Fortunately, the battle is still open and the CKB platform allows him to improve his score by pushing new commits to the GitHub repository, so he decides to work on the battle for a couple of days and then push his work to the GitHub repository. After checking his score again, he sees that he is now in the 1st position and moreover he obtained a badge for being the first to reach 100 points in the battle and now both students and professors can see this badge when they visit Donald's profile.

Closing a battle

When the deadline for the battle created by Scrooge is reached, all participants are notified that the battle is closed and that they can't push new commits to the GitHub repository. Scrooge is notified that the battle is closed and he can now evaluate the code of each team and assign extra points for the clarity of the comments and the code, as he decided when he created the battle. After the evaluation, the final rank of the battle is available to all participants and the students are notified that they can now see the final rank of the battle.

Closing a tournament

Chip decides to close the "Strings Operations" tournament, even if the deadline is not reached yet. In order to do so, he logs in to the CKB platform and he closes the tournament. All participants are notified that the tournament is closed and that they can't join it anymore, in the end, the CKB platform makes the final rank of the tournament available to all participants.

Accessing the scores of the players

Huey wishes to enroll to the class of Advanced Algorithms and Data Structures held by professor Pippo, so he applies for the class. Pippo, who wants to make sure that Huey is a good student, comes to know that Huey is a very active user of the CKB platform and he decides to check his profile. He sees that Huey has a very high score in the "Strings Operations" tournament and that he has a badge for being the most active user of the platform and he also notes that Huey is involved in more than one tournament simultaneously. Thanks to the CKB platform, Pippo has now a complete overview of Huey's skills and he can decide whether to accept his application or not.

2.1.2 Class Diagram

2.1.3 State Diagrams

The following state diagrams describe the life cycle of the main entities of the system. Moreover, they also specify the sequence of states that an object goes through during its lifetime in response to stimuli from the environment. We want to focus on the events that cause a transition from one state to another and the actions that result from a state change.

Tournament

After an educator creates a tournament, it is both in the *registration open* and *tournament open* states.
In the *registration open* state, students can join the tournament, while in the *tournament open* state, educators with the right permissions can create battles within the tournament, and that leads the tournament to the *battling* state.
When the deadline for the registration is reached, the tournament moves to the *registration closed* state and no more students can join it.
When the deadline for the registrations is reached, no more students can join the tournament and it moves permanently to the *registration clcsed* state.
During the *battling* state educatos can start multiple parallel battles and, if and only if all battles are ended, the educator can finally close the tournament.

Battle

The battle evolves in a linear way, starting from the *registration opem* immediately followed by the *registration closed* state.
After the registration deadline is reached, the github repository of the battle is created and thus the battle moves to the *coding* state, allowing the students to fork the repository and start working on the battle.
When the deadline for the battle is reached, the educators can start evaluating the code of the students, if previously enabled (*consolidation* state).
After the evaluation is completed, the battle can be closed and the final rank is available to all participants.

Score evaluation

The score evaluation is a process that is triggered by the end of a battle and it is composed by multiple steps.
First, three aspects can be automatically evaluated: functional aspects (the higher the better, +), timeliness (the lower the better, -) and quality level of the sources, extracted through static analysis tools (+).

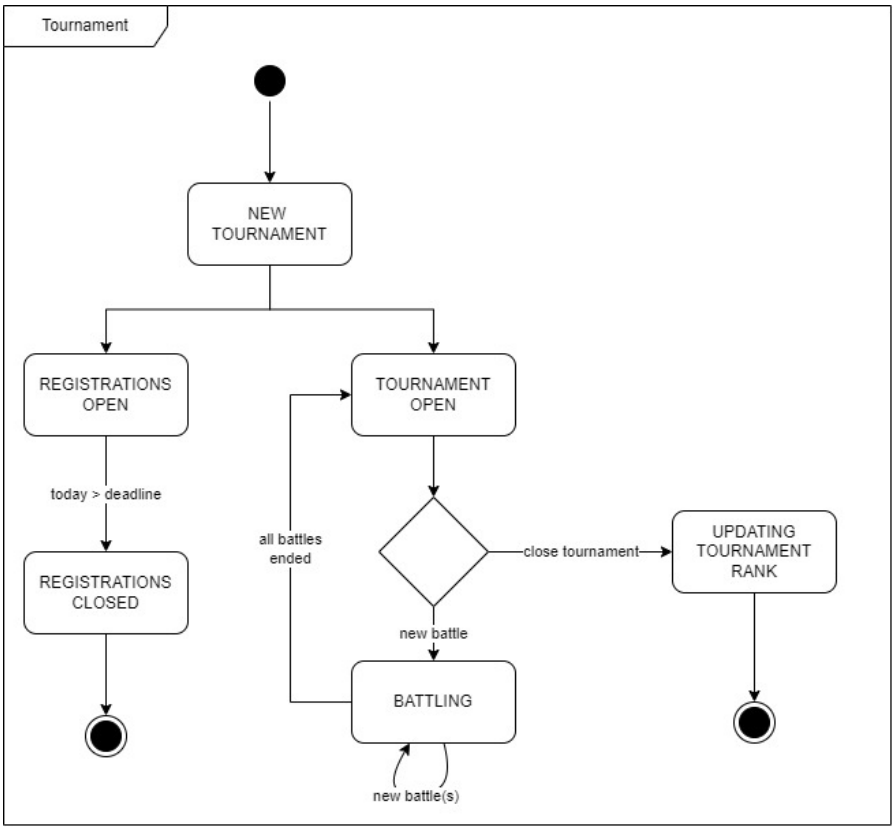

Figure 2.1: Tournament state diagram

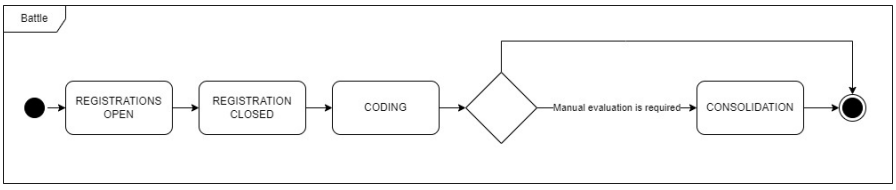

Figure 2.2: Battle state diagram

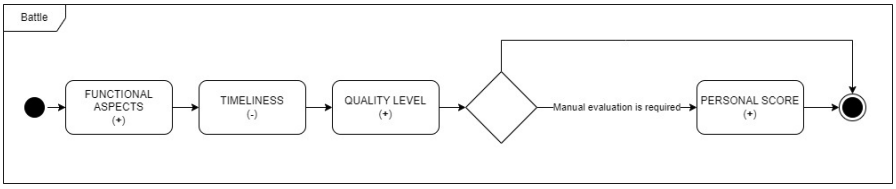

Figure 2.3: Score evaluation state diagram

Finally, if the educator enabled the manual evaluation, he can assign extra points.

2.2 Product Functions

2.3 User characteristics

2.4 Assumptions, Dependencies and Constraints

2.4.1 Domain Assumptions

2.4.2 Dependencies

2.4.3 Constraints

- The software must follow local laws and rules, especially when it comes to handling user data, like letting users access their data when they want.
- The software should only collect the data it really needs, like just the user's email address.
- To keep users' important info safe, like passwords, it must be stored in SHA256 encoding in the database.
- When choosing external APIs, especially those that are crucial for it to work properly, we should pick the ones that are the most dependable and always available.

3. Specific Requirements

3.1 External Interface Requirements

In the following, the interfaces, both hardware and software, of the system are described.

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

To use the CKB platform, both the Educators and the Students need an electronic device connected to the Internet, like a computer, a tablet or a smartphone.
As the platform's primary functionality is closely tied to coding activities, it is expected that users will predominantly employ personal computers to access an Integrated Development Environment (IDE). Consequently, the platform's interfaces have been optimized for use on computer screens.

3.1.3 Software Interfaces

Since the platform is web-based, it is compatible with all the major operating systems, as long as they have a modern browser installed.

3.1.4 Communication Interfaces

The system requires a stable internet connection to work properly. The backend of the system will expose a unified RESTful API to communicate with all clients.
Furthermore, he system relies on various external interfaces accessible via uniform web API. These services are:

- **GitHub API:** to create and manage repositories and to retrieve the code of the students and to authenticate users.
- **Mail API:** to send emails to the users to notify them about events.

3.2 Functional Requirements

3.3 Performance Requirements

To guarantee a good user experience, the system must:

- Make sure the backend can grow as needed, respond quickly to changes, and balance the workload effectively.
- Be protected against DDoS attacks to keep the system safe and stable.
- Create a user-friendly, responsive front-end. It should handle well even when the internet isn't great, so users have a smooth experience.
- Send push notifications really quickly, so users don't even notice the delay.

3.4 Design Constraints

3.5 Software System Attributes

3.5.1 Reliability

Since some functionality of the system relies on external APIs, though the system should not completely fail because of failure in one of those.
It's also important to avoid data loss through redundant storage methods.

3.5.2 Availability

In the event of an unplanned system downtime, all features should be restored as quickly as possible to minimize any inconvenience. To prevent such occurrences, it is crucial for the CKB platform to have a reliable infrastructure, including redundant servers, to ensure continuous operation. The aimed availability of the system is 99.5%, which means that the system can be down for a less than two days per year.
The system should also be able to handle a large number of concurrent users.

3.5.3 Security

Users of the system have distinct privileges according to their roles (student / educator), determined during the login process.
All data and information transferred and stored within the system are secured through robust encryption methods, such as HTTPS, ensuring data privacy and security.

3.5.4 Maintainability

The source code and associated documentation must include clear comments and should be consistently maintained. During the design and development phases, emphasis should be placed on achieving modularity, minimizing coupling, and ensuring high cohesion between components. This is especially crucial for both the front-end and back-end, allowing developers to make updates to the back-end seamlessly without causing any disruptions or noticeable changes for users.
To avoid inconvenience in solving any type of problem (e.g. server downtime), maintenance services are notified to all users with an advance notice of at least 36 hours.

3.5.5 Portability

Due to the fact that the CKB platform is a distributed system, and it doesn't rely on a specific hardware or software, it can be used / accessed in multiples way.

Summary of Non-Functional Requirements

4. Formal Analysis Using Alloy

5. Effort Spent

Team

Topic	Time
Division of work	2h

Table 5.1: Effort Spent during team meetings

Tommaso Pasini

Topic	Time

Table 5.2: Effort Spent by Tommaso Pasini

Elia Pontiggia

Topic	Time
Scenarios	1h
State diagrams	2h
Specific requirements	1h30m
LaTeX document setup and configuration	2h

Table 5.3: Effort Spent by Elia Pontiggia

Michelangelo Stasi

Topic	**Time**

Table 5.4: Effort Spent by Michelangelo Stasi