Accademic Year: 2023-2024

POLITECNICO
MILANO 1863

Requirements Analysis and Specification Document

CodeKataBattle

SOFTWARE ENGINEERING 2

PROFESSOR DI NITTO ELISABETTA

CONTI ALESSANDRO CODICE PERSONA: 10710583 MATRICOLA: 252665

DI PAOLA ANTONIO CODICE PERSONA: 10717589 MATRICOLA: 956038

Contents

1. INTRODUCTION

1.1. Overview

The following RASD aims at providing an overview of the project CodeKataBattle. This document will guide the reader in understanding the purpose of the project, on which grounds it is set and in which environment it will live. Precisely, it will illustrate both informally and formally which are the goals and how these may be reached by guaranteeing certain functional and nonfunctional requirements, while certain phenomena hold in the world. The document will provide a baseline for the project's planning and estimation and will support the software's evaluation. Finally, it will be the starting point for the software's review and update. The document is addressed to all those who intend to benefit from the service and who may be involved in its development.

1.2. Purpose

The purpose of our project, named CodeKataBattle (abbreviated as CKB), is to provide a digital platform that enables students to develop and refine their skills in software development through a collaborative experience. The CKB platform allows students to practice in a programming language of their choice. The exercises offered on this platform are created by educators and require students to complete the code or parts of it, adding the missing components they deem appropriate. The code is then executed and subjected to specific tests created by educators to verify its correctness.

Exercises are organized into tournaments by educators, within which 'battles' are held to evaluate the performance of the students. This process culminates in the creation of performance rankings among all students involved in the battles and tournaments. In this way, CKB encourages friendly competition and the growth of students' skills in the field of software development.

1.2.1. Goal

[G1] Educator creates a tournament and within it, he creates one or more battle.

[G2] Student registers for a tournament and will receive all notifications related to it.

[G3] Student joins a battle within a tournament they are register for and invite some friends to partecipate with them, thus creates a team for the battle.

[G4] Student receives an evaluation for the code he pushed to their GitHub repository.

[G5] Educator reviews all the evaluated code, and if they deem it necessary, he can modify the evaluation they have received.

[G6] Educator closes a tournament and the platform publishes the ranking.

[G7] Both students and educators check the update ranking of the battle and the tournament.

1.3. Scope

Following the definition originally proposed by M. Jackson and P. Zave in 1995, we will try to clarify the scope and the application range of the System by identifying the so called World and Machine phenomena.

The Machine is the System to be developed, in this case a software, while the World, also known as environment, corresponds to the part of the real world that is affected by the System.

The World and the Machine are associated with different types of events which we are going to describe and detail. World and the Machine are associated with different types of events which we are going to describe and detail.

1.3.1. World phenomena

Phenomena events that take place in the real world and that the machine cannot observe.

[WP1]. Educator wants to create a tournament
[WP2]. Educator wants to create a battle
[WP3]. Student wants to join a tournament
[WP4]. Student contacts peers participating in the tournament to form a team
[WP5]. Students fork the GitHub repository
[WP6]. Students and Educator wants to subscribe into the platform

1.3.2. Shared phenomena

1.3.2.1. Controlled by the machine and observable by the world

[SP1]. The platform assigns scores to groups to create a competition rank
[SP2]. Students are notified when a new tournament are creates
[SP3]. All student that are subscribed on a tournament are notified of all upcoming battles creation
[SP4]. The platform creates a GitHub repository containing the code Kata and then sends the link to all students who are members of subscribed teams
[SP5]. When the battle ends the system evaluates the student automatically

1.3.2.2. Controlled by the world and observable by the machine

[SP6]. Students join into a tournament
[SP7]. Educator creates a tournament
[SP8]. Educator creates a battle
[SP9]. Educator gives permission to other ones to create battle in the tournament
[SP10]. Group of student join into a battle
[SP11]. Groups deliver their solution to the platform
[SP12]. Students set up the automated workflow in GitHub so that informs the CKB as soon as students push a new commit.
[SP13]. Each push before the deadline triggers the CKB platform, which pulls the latest sources, analyzes them, and runs the tests on the corresponding executable to calculate and update the battle score of the team
[SP14]. At the end of the battle, optionally the educator evaluates the student by overwriting the result that the platform returned

1.4. Definitions, Acronyms, Abbreviations

1.4.1. Definitions

- Commit: Commits are the core building block units of a Git project timeline. Commits can be thought of as snapshots or milestones along the timeline of a Git project. Commits are created with the `git commit` command to capture the state of a project at that point in time.
- Fork: A fork is a new repository that shares code and visibility settings with the original “upstream” repository.
- Code kata: Code kata contains the description of a battle and the software project on which the student will have to work, including also test cases and build automation scripts
- Test Case: A test case is a singular set of actions or instructions that the Educator wants to perform to verify a specific aspect of the project pushed by the students. If the test fails, the result might be a software defect that students might haven’t found.
- Upload: Upload in which the educator sends to the platform database the code kata for a specific battle.
- Repository: A Git repository is the .git/folder inside a project. This repository tracks all changes made to files in your project, building a history over time. Meaning, if you delete the .git/folder, then you delete your project’s history.
- Branch: In Git, a branch is a new/separate version of the main repository. Branches allows users to work on different parts of a project without impacting the main branch. That main branch is the one seen as the default one.
- Push: The `git push` command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo.
- Pull: The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.

1.4.2. Acronyms

- CKB: CodeKataBattle
- CK: CodeKata
- GH: GitHub
- GHA: GitHub Action
- GDPR: General Data Protection Regulation

1.4.3. Abbreviations

- [Gn]: the n-th goal of the system
- [WPn]: the n-th world phenomena
- [SPn]: the n-th shared phenomena
- [UCn]: the n-th use case
- [Sn]: the n-th scenario
- [DAn]: the n-th domain assumption
- [Dn]: the n-th dependency
- [Cn]: the n-th constraint
- [Rn]: the n-th functional requirement
- Repo: Git repository

1.5. Revision history

1.6. Reference documents

This document is strictly based on:

- The specification of the RASD and DD assignment of the Software Engineering 2 course, held by professor Matteo Rossi, Elisabetta Di Nitto and Matteo Camilli at the Politecnico di Milano, A.Y 2022/2023.
- Slides of Software Engineering 2 course on WeBeep.
- Official link of CodeKata: http://codekata.com/.

1.7. Documents Structure

The rest of the document is organized as follows:

- *Overall Description* (Section 2) contains an in-depth description of the domain under analysis, following the Jackson-Zave approach used in the Scope paragraph in Section 1. Class Diagrams will be hereby provided to formalize all the actors involved in the System and their relations, while their needs and scope as Stakeholders will be analyzed later in the section. Key functional requirements will then be listed, together with all the Domain Assumptions that can be used to entail each Goal.
- In the *Specific Requirements* (Section 3) functional requirements are associated to use cases and Sequence Diagrams to clarify processes and interactions between users and the CKB. Nonfunctional requirements are also included here, together with external constraints (mainly regulations) imposed on the System.
  Developers can also find requirements regarding External Interfaces, with a focus on Hardware and Software Interfaces. Finally, a brief overview of the properties that the CKB will have to guarantee is provided.
- *Formal Analysis* (Section 4) uses Alloy to generate a formal model of two distinct parts of CodeKataBattle, the identification and requests.

2. OVERALL DESCRIPTION

2.1. Product perspective

In this section we will explain in more detail the various World, Machine and shared phenomena. We will also provide some of the descriptions with State and Class Diagrams.

2.1.1. Scenarios

[S1]. Educator creates a new tournament

Greg and Paul are two computer science teachers who want to assess their students' programming skills. While Greg was browsing the internet, he discovered that CKB platform could be suitable software for this purpose. As a result, he decided to register on the platform and asked Paul to do the same. To complete the registration, he had to fill out a form with his personal information and received a confirmation via email once the registration was completed.

Subsequently, Greg logged into the platform by entering the email address and password used during registration on the login page. After Greg logged into the application, he created a new tournament, granting Paul permission to create challenges within the newly created tournament.

Finally, Greg asked all his students to register on the platform as students and join in the tournament he had created.

[S2]. Educator creates a battle

John has created a tournament and now wants to add a new battle to it. To do this, he has prepared a detailed description of the activities to be performed, has set up a code with some parts to be completed, established evaluation criteria, chosen the group size, and set the registration deadline. Now, John can access his profile to actually create the challenge on the platform. From the main page, he selected the tournament he just created and filled out the form with the previously prepared information. At this point, the platform will notify all the students registered for the tournament, and those interested have time until the deadline to form a group and register.

[S3]. Student joins a tournament

Ted wants to improve his programming skills, and a friend suggests trying the CKB platform. Ted trusts his friend and decides to register as a student on the platform. Now that he has created his profile, he can access the platform and start searching for a tournament to join.

From his homepage, Ted can view and search for all the available and active tournaments. In case he doesn't find a tournament that meets his needs or there are no active tournaments, the platform will inform him by sending an email when a new tournament is created.

When Ted finds a tournament suitable for his needs, all he has to do is register before the established deadline, and he can start improving his programming skills.

[S4]. Student joins a battle

Ann, along with her friends Liz and Eddie, received a notification from the CKB platform about the creation of a new battle in the tournament organized by Mr. Smith. They decided to join as a team, and Ann initiated the process. To do so, Ann logged into the platform and, from Mr. Smith's tournament page, found the newly created competition, registered, and invited her two friends to join her to form a group.

Once the registration deadline had passed, they received an email from the platform containing a link to their GitHub repository created specifically for

this battle. Eddie, being the first to view and open the email, accessed the GitHub repository and, following the instructions, forked the repository and modified the GitHub Actions so that every new commit made by her or her friends would be visible to the CKB evaluation platform.

[S5]. Student push a commit in the main branch

Mary is participating in a battle, and after making a change to the project in a secondary branch and ensuring that everything works correctly, she decides to merge her branch with the main branch. After confirming that the branch merge was successful, Mary makes a commit that activates the GitHub Actions, allowing the CKB platform to evaluate the changes made.
The proposed solution is assessed positively, and this enables Mary to achieve a score of 95/100. If she wants to further improve her score, she can make additional code changes by making additional commits before the battle's deadline.

[S6]. Last battle ends and educator closes the tournament

Tom is an educator who has created a tournament on the CKB platform. He has already assigned six battles, and the seventh has just concluded.
All the students who participated in the seventh battle were automatically notified of its completion and can view a draft of their scores and the related ranking. However, Tom noticed that Alice used a highly maintainable solution by implementing some design patterns. He wishes to reward her with an extra 5 bonus points and proceeds to modify the scores. Once the interval for score modifications has ended, a new email is sent to all participants with their final scores and rankings.
At this point, the results of this battle are added to the overall results of the students in the tournament, thus affecting the general ranking.
Tom decides that seven battles are more than sufficient for this tournament and closes it, allowing the platform to automatically send an email to all registered students to inform them of the tournament's closure along with the final rankings and scores.

2.1.2. Domain class diagram

Figure 1 illustrates the High-Level Class Diagram associated with CKB. This diagram encompasses all the components within the system's domain and illustrates the interactions among these elements. The subsequent section provides detailed descriptions of the key components of the diagram to enhance comprehension of the domain.

- We identify the User class, which represents the users who interact with the application. This class is divided into two subclasses that delineate the two types of users in the system: the Student and the Educator.
- It can be noted that the Educator class is linked to the Tournament class because it is created by the educator, and the Student class is also connected as students actively participate by registering for the tournament.
- The Tournament class is associated with the Battle class since multiple battles between students take place within a tournament. Students enroll in battles as teams, hence the Team class depicted in the diagram.
- Given that each battle involves a task to be completed, code to be filled in, test cases to pass, and deadlines, all this information, entered by the educator who created the battle, is stored in the CodeKata class. This class is linked to the Battle class, highlighting the connection between the programming challenge specifications and the actual battle event.
- The evaluation of a student is stored in the Evaluation class, which is linked to the ongoing battle. Each time a new code is added for evaluation, the Battle class

retrieves the necessary information from the CodeKata class, assesses the student's work, and saves the evaluation in the Evaluation class.

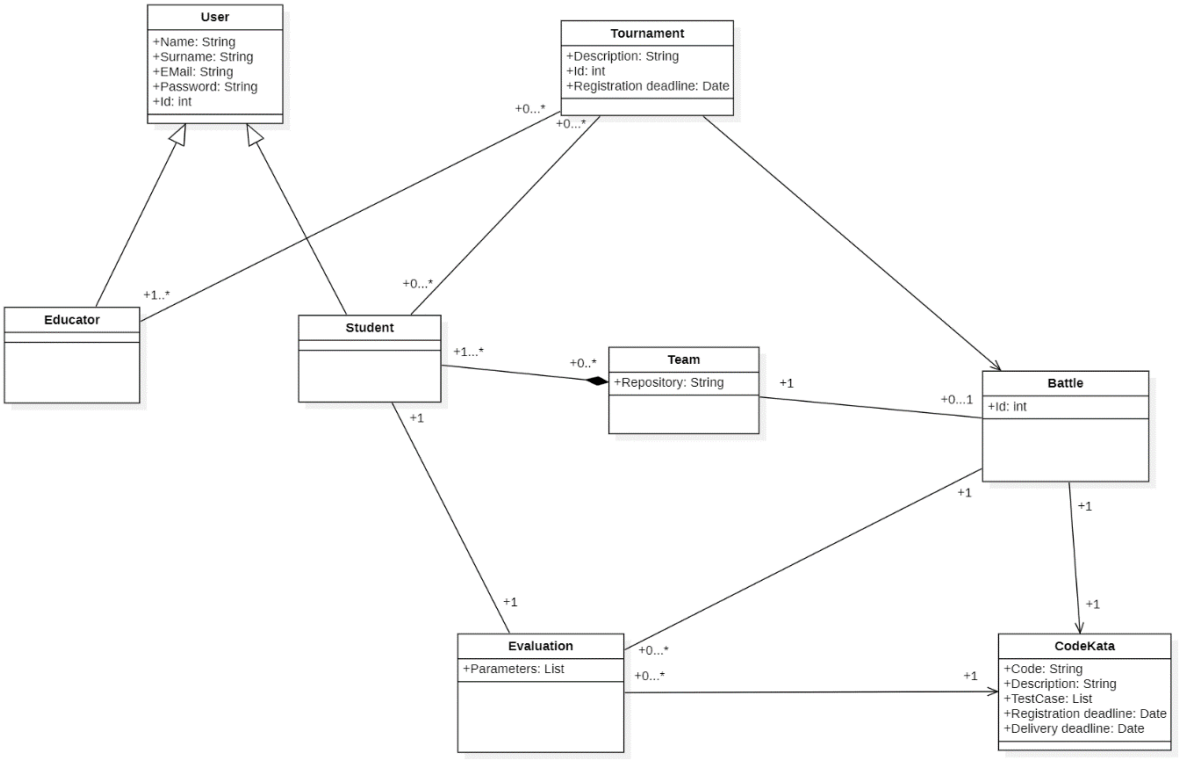

Figura 1 Domain Class Diagram

2.1.3. Statecharts

2.1.3.1. Tournament state diagram

a. *Registration*: In the registration phase, all the operations that occur from the moment the Educator begins entering the data to create the tournament until the deadline for student enrollment are included.
b. *On Going*: The system enters this state if there is at least one participant in the tournament when the enrollment deadline has expired and it persists until a battle is initiated.
c. *Battle*: This state handles the management of battles, from the moment an educator begins entering data for the first battle until the tournament is completed.
d. *Ended*: This state is responsible for concluding the tournament, which can occur either because no student has enrolled or because an overseeing educator decides to close it.

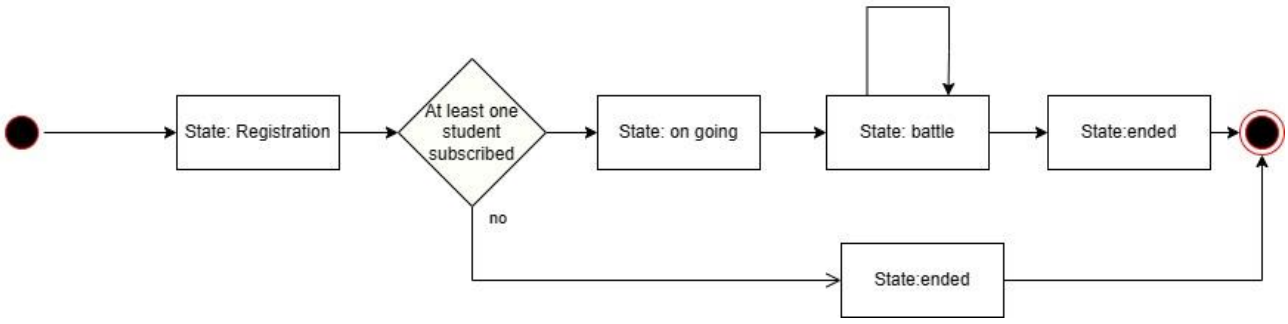

Figura 2 Tournament State Diagram

2.1.3.2. Battle state diagram

a. *Registration*: In the registration phase, all the operations that occur from the moment the Educator begins entering the data to create the battle until the deadline for student enrollment are included.
b. *On Going*: The system enters this state if there is at least one participant in the battle when the enrollment deadline has expired and it persists until the battle is ended.
c. *Ended*: This state is responsible for concluding the battle, which can occur either because no student has enrolled or because the battle deadline occurs.

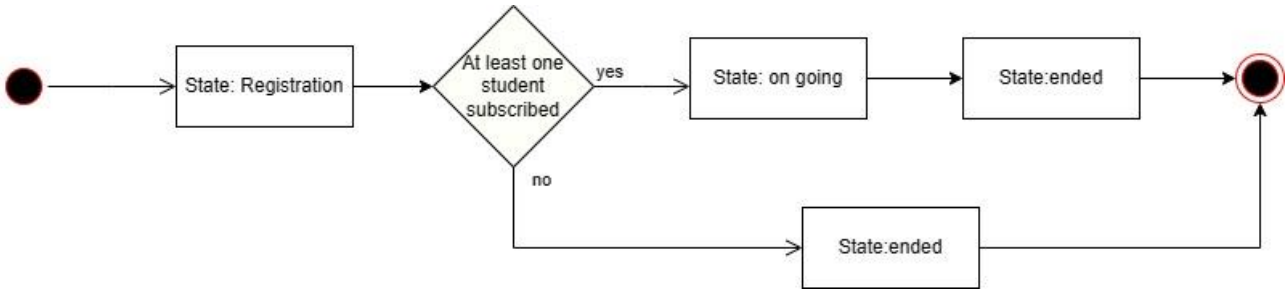

Figura 3 Battle State Diagram

2.2. Product functions

This section is devoted to analyzing the Value Proposition that CodeKataBattle provide to its users. The presented functionalities are grouped according to the user they address.

2.2.1. Sign up and Login

These functions will be available to all users.
The registration functionality allows users to sign up for the system. Each user will need to choose whether to register as a student or an educator, and they will need to provide an email address, a password, and some personal information. Once the registration is completed, the system will send a confirmation email, officially enrolling the user in the CKB platform.
The login functionality allows registered users to access the platform and start using it.

2.2.2. Manage a tournament

One of the main functions of the platform is to comprehensively manage tournaments. The platform allows educators to create tournaments and involve their colleagues as collaborators for their management. After creating a tournament and setting the registration deadline, the platform will send an email notification to all registered students, informing them of the creation of a new tournament and offering them the opportunity to register.

Furthermore, when an educator creates a new battle or when the overall ranking is updated, the platform will send notifications to all participating students, keeping them informed about any changes that occur within the tournament to which they are registered.

2.2.3. Manage a battle

One of the main functions of the platform is to comprehensively manage all aspects of battles within a specific tournament. The platform allows educators to create a battle, including a detailed description of the exercise to be completed, a code with parts to be filled in by students, evaluation criteria, the group size, and a registration deadline. After creating the battle, the platform will send a notification to all students registered for the tournament, allowing them to enrol. When the deadline is exceeded, the platform will automatically create GitHub repositories containing the exercise to be completed. When a student activates the GitHub Actions, allowing the platform to receive all the commits made by them, it will evaluate the solutions according to the criteria chosen by the educator.
At the conclusion of the battle, the platform will generate a ranking based on the evaluations of all students and notify the results. Subsequently, within a new deadline, the educator will have the opportunity to modify the scores assigned to individual students. After the deadline for modifying grades has passed, the platform will add the results obtained in this battle to the scores from previous battles, altering the overall tournament leaderboard.

2.3. User characteristics

CKB addresses two types of users: Student and Educator.

2.3.1. Student

A Student is a user eager to hone his skills in software development and decides to enrich his education by registering on the CKB application.
After successfully registering on the application, the Student has the opportunity to immerse himself in challenging tournaments. Once registered in these tournaments, he can engage in battles against other Students equally eager to improve their skills. In this competitive environment, the user has the opportunity to compare his results with those of his peers, creating a dynamic and collaborative environment for learning and developing software development skills.

2.3.2. Educator

An Educator is a user who shares his or her skills in software development with other users eager to improve their skills.
After registering, the Educator has the ability to create tournaments, organising programming battles within them in which registered Students can participate to hone their skills. The Educator oversees the tournament by monitoring the codes submitted by the Students to the platform, providing additional evaluation in addition to the automatic evaluation. This process creates a collaborative environment where students can receive direct and targeted feedback to further improve their software development skills.

2.4. Assumption, Dependencies and Contraints

2.4.1. Domain Assumption

[DA1]. Users must have a valid email address
[DA2]. Students must have a GH profile
[DA3]. Students need to know how to access GitHub, how to fork a repository and how to enable GitHub Actions
[DA4]. Each user will create only 1 account

[DA5]. Personal data given by User during the registration process are assumed true

[DA6]. User must have an internet connection

[DA7]. Educator must upload the correct CK

2.4.2. Dependency

[D1]. The CKB will make use of the GHA service offered by GitHub user account

[D2]. The CKB will access GitHub APIs to create new repositories and add Students

2.4.3. Constraint

[C1]. The CKB must abide to the GDPR regulation with regards to the treatment of personal data.

[C2]. The CKB must be implemented as a web site

3. SPECIFIC REQUIREMENTS

3.1. External interface

3.1.1. User interface

3.1.2. Hardware interface

3.1.3. Software interface

3.1.4. Communication interface

3.2. Functional requirements

In the following are specified all the requirements that the system has to fulfill. In order to work properly, the system should:

[R1]. System allows students to sign up
[R2]. System allows educators to sign up
[R3]. System allows students to login
[R4]. System allows educators to login
[R5]. System allows educator to create a new tournament and to add all the needed information
[R6]. System allows educator to grant the permission to other colleagues to create battle in a specific tournament
[R7]. System allows educator to create a battle inside a particular tournament by adding CK and the deadlines
[R8]. System allows educator to see rankings of each tournament, which he created or in which he has been nominated collaborator
[R9]. System allows educator to see ranking to see ranking of a specific battle…
[R10]. System allows educator to modify the evaluation manually of a specific group in a battle…
[R11]. System allows educator to close a tournament
[R12]. System notifies students about the creation of a new tournament
[R13]. System notifies students about the creation of a new battle in a specific tournament in which they have subscribed
[R14]. System notify student about publication of final ranking of a specific battle…
[R15]. System notify student about publication of final ranking of a specific tournament…
[R16]. System allow student to join a tournament
[R17]. System allow student to join a battle
[R18]. System allows students to see rankings of each tournament, which they are sub scripted
[R19]. System allows students to see ranking to see ranking of a specific battle…
[R20]. System, at end of the registration period of a tournament, create a GH repo and send invitation to all member of the group
[R21]. System automatically evaluates the code in the main branch of the repo after each commit in it
[R22]. System allows students to search tournament by key words

3.2.1. Use cases diagram

3.2.2. Activity diagram

3.3. Performance requirements

3.4. Design constraints

3.4.1. Standards compliace

3.4.2. Hardware limitations

3.4.3. Any other constraint

3.5. Software system attributes

3.5.1. Reliability

3.5.2. Availability

3.5.3. Security

3.5.4. Maintainability

3.5.5. Portability

4. FORMAL ANALYSIS USING ALLOY

4.1. Signatures

4.2. Facts

5. EFFORT SPENT

6. REFERENCES