Requirement Analysis and Specification Document

Christian Rossi
Kirolos Sharoubim

Academic Year 2023-2024

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3
1.3 Definitions, acronyms and abbreviations . . . 5
1.3.1 Definitions . . . 5
1.3.2 Acronyms . . . 5
1.3.3 Abbreviations . . . 5
1.4 Revision history . . . 5
1.5 Reference documents . . . 5
1.6 Document structure . . . 6

2 Overall description **7**
2.1 Product perspective . . . 7
2.1.1 Scenarios . . . 7
2.1.2 Domain class diagrams . . . 7
2.1.3 State diagrams . . . 7
2.2 Product functions . . . 8
2.2.1 First function . . . 8
2.3 User characteristics . . . 8
2.3.1 Student . . . 8
2.3.2 Educator . . . 9
2.4 Assumptions, dependencies and constraints . . . 9

3 Specific requirements **10**
3.1 External interface requirements . . . 11
3.1.1 User interfaces . . . 11
3.1.2 Hardware interfaces . . . 11
3.1.3 Software interfaces . . . 11
3.1.4 Communication interfaces . . . 11
3.2 Functional Requirements . . . 11
3.3 Performance requirements . . . 11

3.4 Design constraints 11
3.4.1 Standards compliance 11
3.4.2 Hardware limitations 11
3.4.3 Any other constraint 11
3.5 Software system attributes 11
3.5.1 Reliability 11
3.5.2 Availability 11
3.5.3 Security 11
3.5.4 Maintainability 11
3.5.5 Portability 11

4 Formal analysis with Alloy **12**

5 Effort spent **13**

6 References **14**

Chapter 1

Introduction

1.1 Purpose

CodeKataBattle is a platform employed by educators to challenge students on code Katas, problems meant to be tackled using a programming language selected by the organizers of the challenge.

Specifically, educators have the capability to establish a code Kata within a particular tournament by defining the following parameters: the problem to be solved (with some test cases), the minimum and maximum number of students per group, and the deadlines for code Kata registration and solution submission.

After the creation of the tournament, students can form groups and start their work on the solution, employing a test-first approach. When the ultimate deadline approaches, the system autonomously calculates the final rankings and unveils the winners.

1.2 Scope

To ensure the proper development of this application, it is crucial to examine all imaginable phenomena linked to the system.

These phenomena can be categorized into three distinct categories: world phenomena (occurring in the external environment beyond the machine's control), machine phenomena (taking place within the machine and beyond the influence of the external world), and shared phenomena (interactions and events involving both the system and the external world).

Regarding CodeKataBattle, we encounter the following external phenomena:

1. The educator generates a code Kata problem description.

2. The educator formulates a series of test cases for the code Kata problem.

3. The students develop a program to solve the code Kata.

4. The educator assigns a personal score based on code quality.

Additionally, the following internal phenomena arise within the system:

1. The system creates a GitHub repository.

2. The system establishes an automated workflow through GitHub Actions to notify CKB about new commits.

3. The system analyzes the repositories at every commit.

4. The system conducts automated evaluation and updates the score according to the following criteria:

   (a) Number of passed test cases.

   (b) Timeliness between the start of the challenge and the last commit on the main branch.

   (c) Static code analysis for security, reliability, and maintainability.

Lastly, the following shared interactions and events emerge:

1. The educator initiates a battle on the platform, including the following steps:

   (a) Uploading the code Kata.

   (b) Specifying the minimum and maximum number of students per group.

   (c) Setting a registration deadline.

   (d) Setting a final submission deadline.

   (e) Configuring additional scoring parameters.

2. Students submit their implementations to the platform via GitHub commits.

3. The educator creates a tournament.

4. The educator grants permissions to other educators to create code Kata within a specific tournament.

5. Students subscribe to specific tournaments.
6. The student joins an already existing group.
7. The student forms a new group and extends invitations to other students.
8. The educator concludes the tournament, and the system notifies the students.

1.3 Definitions, acronyms and abbreviations

1.3.1 Definitions

Code Kata: adaptation of the concept of karate katas, where you repetitively refine a form, to the realm of software development, fostering iterative practice and improvement.
Test-first approach: software development process relies on the transformation of software requirements into test cases before the software is completely developed, and it involves monitoring the entire software development by iteratively testing the software against all these test cases.

1.3.2 Acronyms

CKB: CodeKataBattle
CK: Code Kata

1.3.3 Abbreviations

1.4 Revision history

Version 1.0 - Release - date TBD

1.5 Reference documents

Document 1 - Presentation about RASD structure
Website 1 - http://codekata.com
Website 2 - https://en.wikipedia.org

1.6 Document structure

This document contains the following elements:

1. **Introduction**: this section offers a general overview of CodeKataBattle and its objectives.
2. **Overall description**: this section provides comprehensive information about the system, including interfaces, constraints, domain assumptions, software dependencies, and user characteristics.
3. **Specific requirements**: this section outlines the system's functionalities through scenarios and use case diagrams.
4. **Formal analysis with alloy**: this section contains the Alloy employed to verify the platform's correctness.
5. **Effort spent**: this section details the amount of time (in hours) contributed by each group member.
6. **References**: this section lists the tools used in the document's development.

Chapter 2

Overall description

2.1 Product perspective

2.1.1 Scenarios

1. *Educator creates a new tournament*
   Description missing

2. *Educator adds a CK to a specific tournament*
   Description missing

3. *Educator closes the tournament*
   Description missing

4. *Student does a commit*
   Description missing

2.1.2 Domain class diagrams

The domain class diagram for CodeKataBattle is presented below, covering all the elements within the system's operational environment and illustrating their interactions.

DIAGRAM HERE

Description of the elements in the class diagram HERE.

2.1.3 State diagrams

Introduction to the state diagrams and why we choose to represent those scenarios.

First diagram

DIAGRAM
    Description of the diagram.

Second diagram

2 DIAGRAM
    Description of the 2 diagram.

2.2 Product functions

Introduction to functions offered and specific description of each one.

2.2.1 First function

2.3 User characteristics

The platform accommodates interaction from two user categories: students and educators. The initial user category engages in Code Kata tournaments, while educators, on the other hand, serve as the organizers of these tournaments.

2.3.1 Student

The student, as an individual aiming to secure victory in a specific tournament, may need to either join an existing team or establish one with fellow students, considering that each tournament has a predetermined group size.

In particular, a student requires access to the following features within the system:

- Student login.
- Personalized user interface.
- Access to the list of available tournaments.
- The ability to join a desired tournament if it's open for registration.
- Capability to create and join groups.
- Invitation of other students to their group.

- Access to the GitHub repository link for a specific tournament.
- Access to both interim and final tournament rankings.

2.3.2 Educator

The educator's role encompasses the creation and management of tournaments, with the potential involvement of other educators designated by the tournament's owner, who is typically the initiating educator. All educators should possess the capability to create code Katas for the tournaments they manage and provide additional points for evaluation.

To facilitate these responsibilities, educators need access to the following features within the system:

- Educator login.
- Personalized user interface.
- Access to a list of tournaments under their management.
- The ability to create new tournaments with customized rules.
- Within a managed tournament:
  - The option to invite other educators to participate.
  - The capability to add new code Katas.
  - The authority to close the tournament.
  - The ability to assign extra points if specified in the tournament rules.
- Access to the GitHub repository for every student in the active tournament.
- Access to both interim and final tournament rankings.

2.4 Assumptions, dependencies and constraints

Chapter 3

Specific requirements

3.1 External interface requirements

3.1.1 User interfaces

3.1.2 Hardware interfaces

3.1.3 Software interfaces

3.1.4 Communication interfaces

3.2 Functional Requirements

3.3 Performance requirements

3.4 Design constraints

3.4.1 Standards compliance

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

Chapter 4

Formal analysis with Alloy

Chapter 5

Effort spent

The table below offers a concise overview of the hours invested by each group member, along with a brief description of their contributions. Dates where the same amount of time was dedicated by all members usually indicates collaborative efforts.

Date	**Rossi**	**Sharoubim**	**Description**
22-10-2023	1	1	Repository setup and file structure
28-10-2023	3	3	First chapter initial content
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
xx-xx-xxxx	-	-	
Total	4	4	-

Chapter 6

References