University of Tartu
Faculty of Mathematics and Computer Science
Software Engineering
System Modeling

Team Members

Carlos Paniagua	B06327
Radomir Sebek	B06334
Chris Willmore	B06338

Project - Mancala
Project Report

Project Report

Team

Chris, Carlos, Radomir

Project Coordination

Members of the team agreed that we should have at least one meeting per week (face to face meeting) in order to review current state of project, discuss problems, make decision and plan future activities. Almost daily we communicate via Skype, but mainly about practical things, smaller decisions and informing each others about progress. Significant decisions were generally made during face-to-face meetings. For code and diagram source control management we used git. Documentation collaboration was done using Google docs.

General Project Flow

At the very beginning of this project all team members learned the Mancala game rules and played already implemented Mancala game on Internet in order to gain ideas for conceptually different user scenarios/stories.

From the onset of our project, we agreed to use Fujaba for the model portion of our application model portion of our project, as it promotes object first development. For the View and Controller we settled on Swing with property change listeners, as we were not familiar with any other framework.

Development began when two scenarios describing application flow were created. Our next step was to derive approximately 24 user stories. At the onset we struggled to create the required 30 stories. We recognized however more stories would become evident during development either through divine inspiration or bugs which exposed story holes.

From the prominent nouns in the initial user stories, we create a handful of pre and post condition object diagrams. At this point we were able to begin development with Fujaba.

The object diagrams were aggregated together to form our first object diagram. We then selected several candidate stories to test the practicality of our architecture. On such story was the simple “User moves a few seeds on his side.” We turned to the verbs in our stories for the method names. The first few methods were developed in group session were we talked through the implementation.

Up until the point, all work was essentially done together during meetings. We felt confident after the project had some mass to began delegating tasks to individual members. At the beginning this took the form of each member creating new user stories and diagrams. Later in development, members specialized in Fujaba, GUI, or documentation.

Development for the last 2 weeks was a constant cycle of Fujaba coding, testing, debugging and code refinement. Many initial design choices were thrown out in favor of cleaner design. One such design choice was refinement of the board structure. The initial design called for “owns” relationship from Player to each of his Houses. To enforce ordering, there was also a “first” and “last” relationship between Player and House, as well as “next” between each house. This design was obviously cumbersome, as it called for too many links. It was only when we became frustrated with the work creating each test, and became more familiar with Fujaba, that we decided to use the ordered relationship feature. This greatly simplified our diagrams and made for easier testing.

Another major design change that we made concerned the Player.play(House) method. The first implementation contained explicit logic for navigation around the board, handling seed transfers, and transferring turn ownership. As we added logic to the method for handling increasing complex operations, the diagram became unworkable. We then decided to segegrate the the various operations into more appropriate classes and methods. Navigation was moved to a new ContainerNavigator, whose sole responsibility is to iterate around the board given the mover and initial house. Seed transfer was moved to a new aptly named Hand class. The Player.play(House) method became instantly more concise and manageable.

The full history of developement and design decissions can be found from SCM commit log (https://github.com/willmore/Systems-Modeling-Mancala/commits/master) and our meeting notes below.

Management information

Task	Person carried out task:
30 user stories	Chris, Carlos, Radomir
40 object diagrams	Chris, Carlos, Radomir
class diagram	Carlos, Chris
15 test cases (15 storyboards)	Chris, Carlos, Radomir
Project report	Chris
guideline to the project	Chris, Radomir and Carlos
meeting reports	Radomir
documented design decisions	Chris, Carlos, Radomir
User manual	Carlos
Documented code	Chris, Carlos
Material for project presentation	Carlos, Radomir
version history	Chris, Radomir, Carlos

Meeting reports

The following are notes on development and design done during group meetings. The notes should be provided a comprehensive view of our application’s creation.

Meeting #1

Date	**26.10.2010**
Place	Liivi 2, IV floor

- Decided to use Fujaba, Java, and Swing becuase they are the technologies we are most familiar with.
- Will hope to use to use SVN (if git continues to make as difficulties with resolving conflicts like for previous hw)
- Next meeting we will review user stories ideas.
- git repository and google docs created

Meeting #2

Date	**31.10.2010**
Time	13.00h
Place	UT library

- We abandon plans to use SVN, as we would prefer to learn git.
- Continue working in google docs files, our plan is to publish them later using git,
- We share our ideas of user stories, agreeing on ~24. At this point of time we have difficulties with defining 30 conceptually different user stories. We agreed that in future days, while working on writing user stories and creating object diagrams we should come up with few more user stories,
- Started creating object diagrams, pre and post conditions diagrams
- Worked on defining pre pre conditions for user stories (board description and notation)
- Comment in class discussion board for some clarification,
- Design Decision: We agreed about primary object design of board. When modeling object diagrams and talking through potential flow we identify the following object relationships: 1. player to his own store, 2. player to his first store, 3. player to his last store, 4. store to its opposite store,
- Agreed that until next meeting we should finish as much as possible user stories and object diagrams.

Meeting #3

Date	**04.11.2010**
Time	18.00h
Place	Liivi 2, IV floor

- Start working on class diagram in Fujaba,
- Design Decision: When modeling seeds in object diagrams we could model every seed separately, but knowing how Fubaja works, it will be time consuming, and not efficient design solution. We decided to model seeds as attributes of House.
- Design Decision: all types of movements (move- no capture, move- capture, move- extra move etc) will be implemented inside play(house : House) : Void method.

- Modelled first user story, Junit test successfully passed

Meeting #4

Date	**08.11.2010**
Time	16.00h
Place	Liivi 2, IV floor

- Added more story boards (test cases).
- Agreed we need few more user stories without GUI, so we can create object diagrams in dia.
- Planned how to implement two more methods, move and create new game.

Meeting #5

Date	**13.11.2010**
Time	15.00h
Place	UT library

- git is working fine :)
- Various minor improvements to class diagram.
- Added logic to Player.Play(House) method so all possible movements are working now.
- Added new storyboard: “Point score via capture”
- Currently all tests are passing.
- Agreed to start with GUI on next meeting

Meeting #6

Date	**14.11.2010**
Time	15.00h
Place:	Raatuse 22, 635

- Player.play(House) method doesnt work, need to improve it
- added one new test case

Meeting #7

Date	**15.11.2010**
Time	09.30h
Place	Liivi 2, IV floor

- Started with GUI, creating game board.
- The Player.play(House) is worked on further as related tests still do not pass.
- Added Game.isGameOver() which determines if game play should continue or not.
- We create Game.startGame() method defined, which initializes all objects necessary to play the game: Players, houses, stores

and turn.
- Game.finalizeGame() method created which scores the game, adds the score to the history, and determines the first player of the next game. This method calls isGameOver().
- 2 more story boards created.
- We now have 9/9 test cases passing successfully

Meeting #8

Date	**16.11.2010**
Time	20.30h
Place	Raatuse 22

- 4 additional user stories and associated object diagrams created.

Meeting #9

Date	**18.11.2010**
Time	16.30h
Place	Liivi 2, IV floor

- Class diagram refinement
- Group work on MVC
- Design Decision: class diagram looks complex at this point of time, it contains several redundant information. By implementing ordered relationship between Houses and House classes we can simplify diagram but keep functionality.(Need to rebuild storyboards for all test cases. now)
- Start working on different types of project documentations

Meeting #10

Date	**20.11.2010**
Time	20.00h
Place	Chris' house

- 30 user stories now completed. There are ones that must be revisited as they may be duplicates or not applicable.

Meeting #11

Date	**25.11.2010**
Time	19.00h
Place	Chris' house

- Continued development on View and Controller.
- Rebuilding test cases, cause many changes were made in class diagram, new logic needs to be implemented in all test cases
- Design Decision: When examining failing tests, we noticed that the History object is not being cleared from previous testl. In

order to solve this problem, we decided to change the History class not to be implemented as a singleton. This prevented leaking “history” between tests. We could have created a clean function, and executed at start of each test . However, we saw that having the classes as singletons did not necessarily contribute to overall design, and was more a hindrance. We decided that hardcoded singletons in general were probably not a great idea (there are failing out of fashion with IOC frameworks) so we removed any other singleton implementations.

Meeting #12

When	**Nov 27**
Place	UT Library

- Design Decision: We discovered that game ending moves weren’t working correctly from Controller because Player.play(House) and Game.finalizeGame() (which controls ending the game) were disjoint. We could have made Controller call Player.play(House) then Game.finalize(), but this felt incorrect as it would created tightly coupled domain level methods. We therefore decided to create Game.play(Player, House) which would first call player.play, then Game.finalize(). We then made Player.play(House) package private as all clients should use Game.play(Player, House). On second look, we didn’t need to take Player as argument to Game.play(), as game knows who turn it is.

- Design Decision: We reviewed whether or not to leave turn abdication logic in Player.play(House), as opposed to including it in the higher level Game.play(House). In order to make turn change in Game.play(House), Game would have to know the last house a seed was placed in order to properly decide who plays next. It was deemed to cumbersome to store this last move information, and so decided to keep turn abdication within the domain of Player.play(House).

Meeting # 13

Date	**Nov 28th**

- Discovered missing a user story to show how the history grows. Added “Third Game Over” user story and test.
- Discovered during gameplay that capture was improperly being executed when opponents house was empty. Created story “Move; no point score; last seed lands in own empty house; capture triggered, opponent empty, no capture” to address this instance.

Meeting # 14

Date	Tuesday Nov 30th.
Time	19.00h
Place	Liivi 2, IV floor

- finalizing project, checking and correcting mistakes in user stories, object diagrams
- final test (15th) created
- working in project report and user manual
- We discovered that we did not implement the randomized part of selecting the first player to move. This was added by creating Turn.choseRandom() which chooses a random player to be on turn using java.util.Random.