Technical Report

Software Engineering 2

Lecturer: Kevin Jackson

Team Anonymous

- Pham Tran Thien – s3312328@rmit.edu.vn
- Phan Thanh Dat – s3311339@rmit.edu.vn
- Nguyen Ho Tan Hung – s3309664@rmit.edu.vn
- Nguyen Tan Tai – s3311341@rmit.edu.vn

Table of Contents

I. Introduction

The purpose of this report is to show the design of the DevFortress game as well as how we applied design patterns and Scrum into managing the project. There are seven parts in this report, namely design, difficulties, applicability of the design patterns, applicability of Scrum project management, changes, additional constrains and tools.

II. Design of the DevFortress game

The following figure shows how our game was designed:

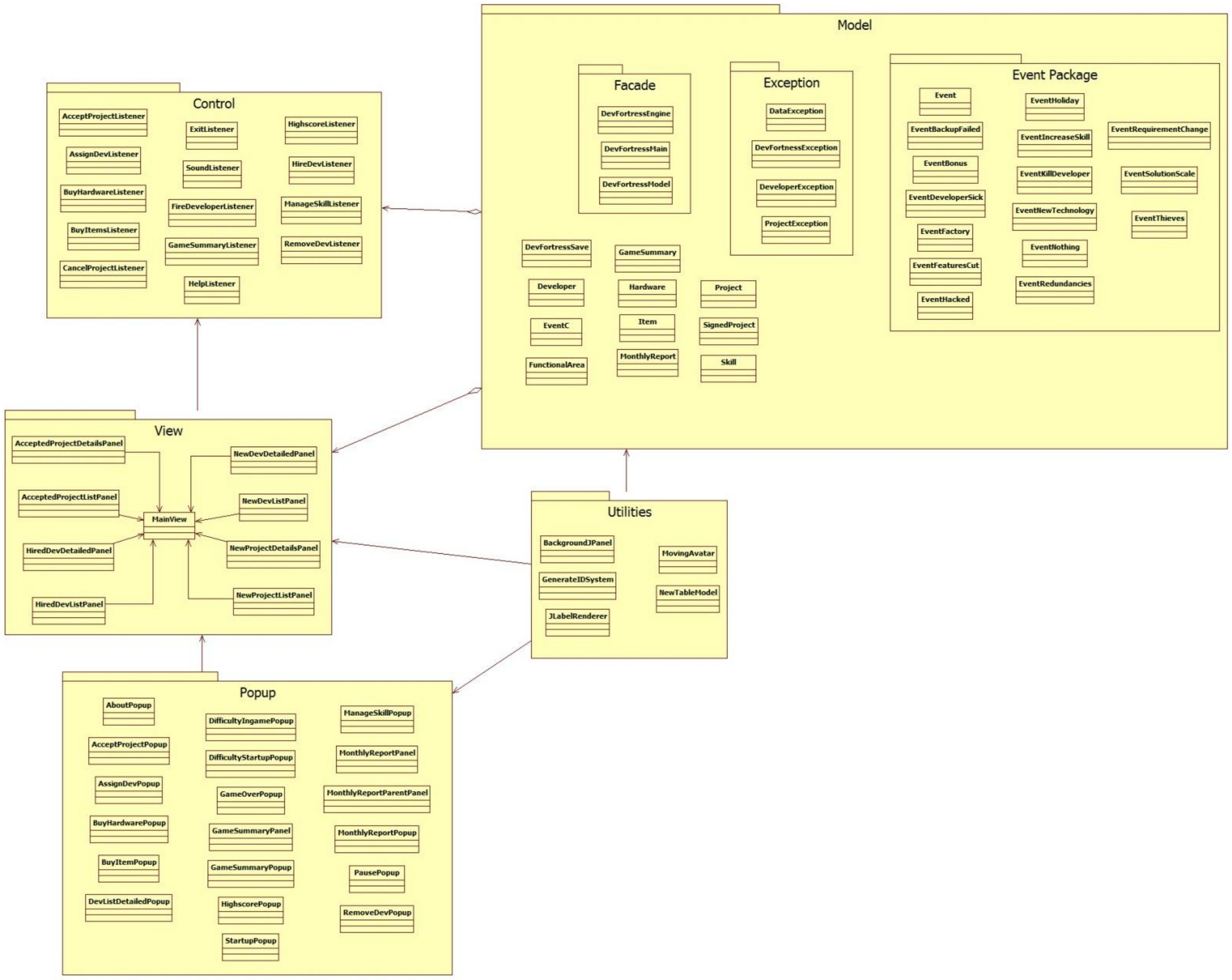


Figure 1: Class Diagram

Observer pattern is applied for the design of the game – which will be discussed in the next section.

The Facade package contains the main classes of the program, namely DevFortressEngine, DevFortressModel and DevFortressMain. Besides that, the Event package handles events happen to developers in the game while all the exceptions in the game will be handled be the package Exception.

Most classes in the Utilities package are used for the GUI, except the GenerateIDSystem is used for automatically creating IDs for the developers. The Popup package is used for showing information in the game like developer's details or project's details.

III. Difficulties

- At the beginning, we used DevFortressMain.model to get the same model object. However, when we ran Unit Test, the methods that included this call DevFortressMain.model caused NullPointerException. The reason is that during Unit Test, the system does not create the DevFortressMain object.
  - ⇨ Solution: In order to solve this problem, we applied the Singleton pattern so that we can get the same model object by using DevFortressEngine.getInstance() instead of DevFortressMain.model.

- Another problem is the team member conflict opinions. There were several ideas about how to implement some methods or how the system works.
  - ⇨ Solution: We organized a meeting to listen to all the opinions and decide to follow the most suitable ones.

- The GUI at the first place was quite complex and not very attractive.
  - ⇨ Solution: At the final week of the first Sprint, all team members had to research for some libraries for the GUI and applied them. Besides that, we added some animations and sound.

- While a member was doing the Unit Test, another member changed the way of saving file of the system. This led to some failed test cased.
  - ⇨ Solution: That team member had to talk to the other one and they had an agreement on the way the system saves and loads data.

- The game is used to be saved by saving DevFortressEngine object. However, when Timer and Clip are implemented in the game, and it is not possible to implement Serializable for these two objects, it caused Serialization Exception as a result.
  - ⇨ Solution: This problem has been solved by creating an object named DevForetressSave and saving this object instead of saving DevFortressEngine object.

IV. Applicability of the design patterns

During the process of develop DevFortress game, we applied the three following design patterns:

a. Factory Pattern

The figure below describes how we applied Factory Pattern for handling events in our game.

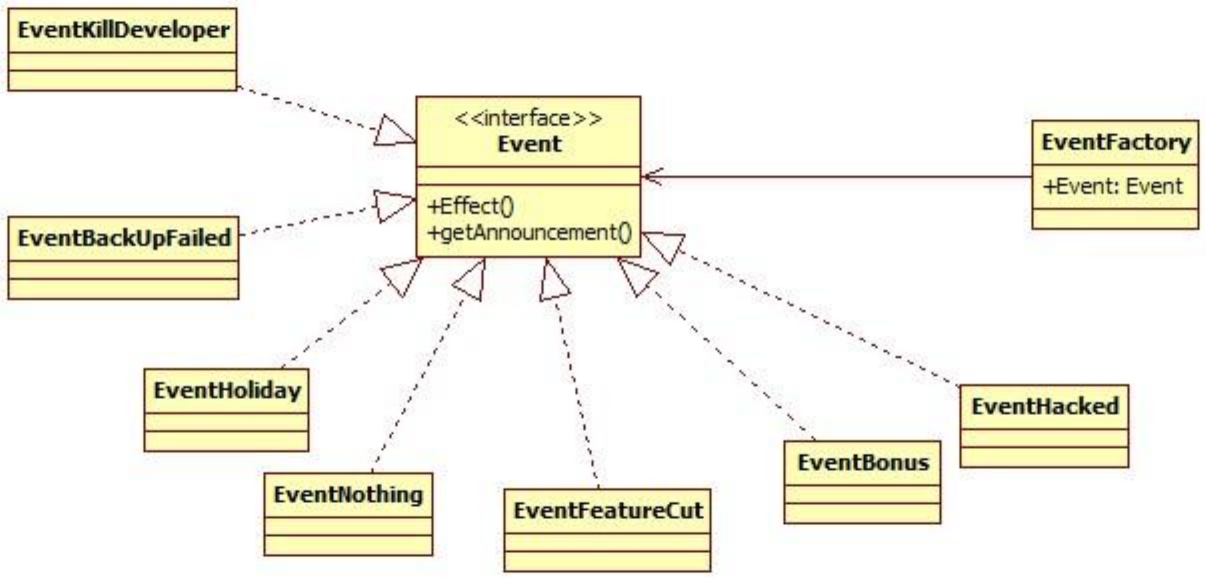
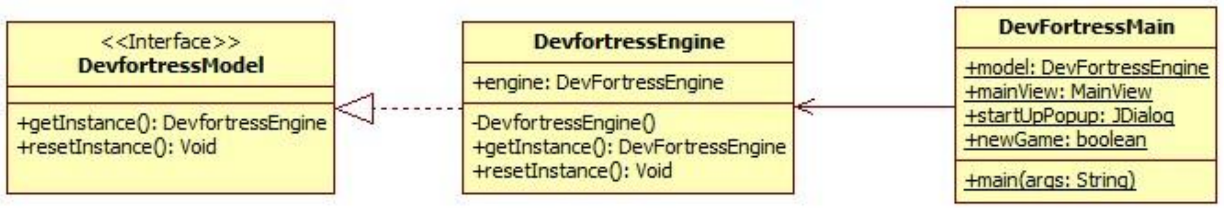


Figure 2: Factory Pattern Diagram

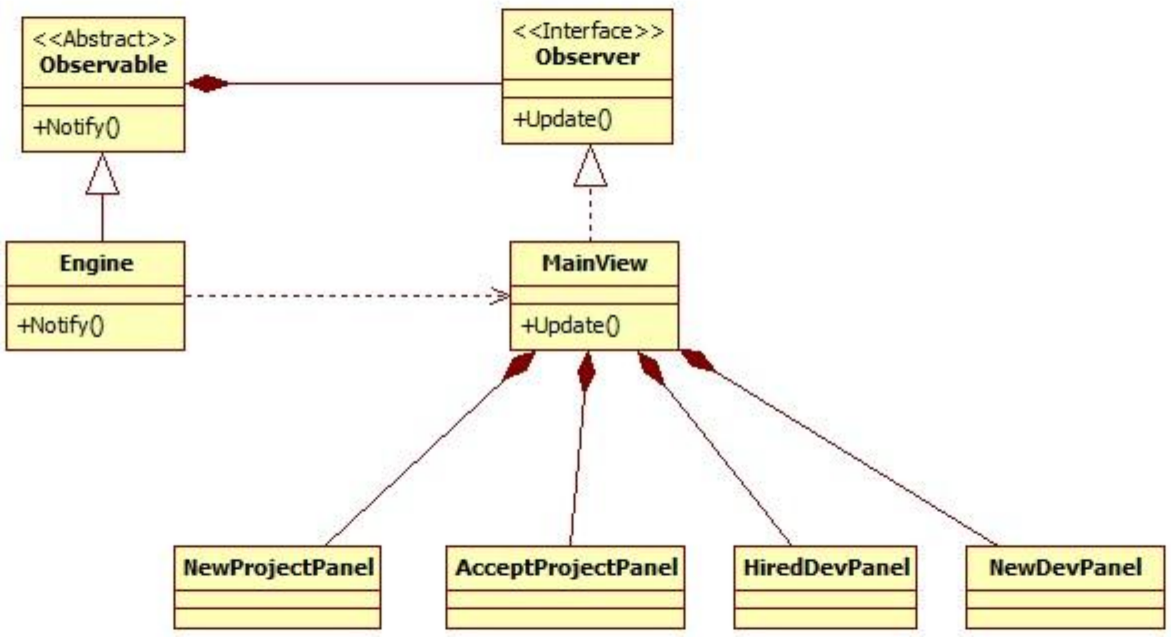
For handling the event action, we use factory design pattern for this situation. The part of event factory will run independently with the system. We have the interface of Event which is implemented by many sub classes of Event such EventKillDeveloper, EventBackUpFailed, etc. and the class of EventFactory where doing the algorithm to get the random Event. Whenever the system needs the random Event, it just needs to get the event from EventFactory class.

b. Singleton Pattern



We applied singleton pattern to our design because it restricts the instantiation of a class to one object. It is quite useful when only one object is created to coordinate actions across the system. Secondly, the object can be accessed by disparate objects throughout a software system and therefore require a global point of access. Additionally, singleton also allows multiple instances in the future without affecting a singleton class's clients.

c. Observer Pattern



We applied Observer pattern in the main design of the game because it provides a powerful way to write classes that are relatively independent from other portions of code, making them re-usable as well as facilitating changes to code without breaking dependencies.

V. Applicability of Scrum project management

The figure below will give a draft projection of how we are going to execute "DevFortress" game based on Scrum.

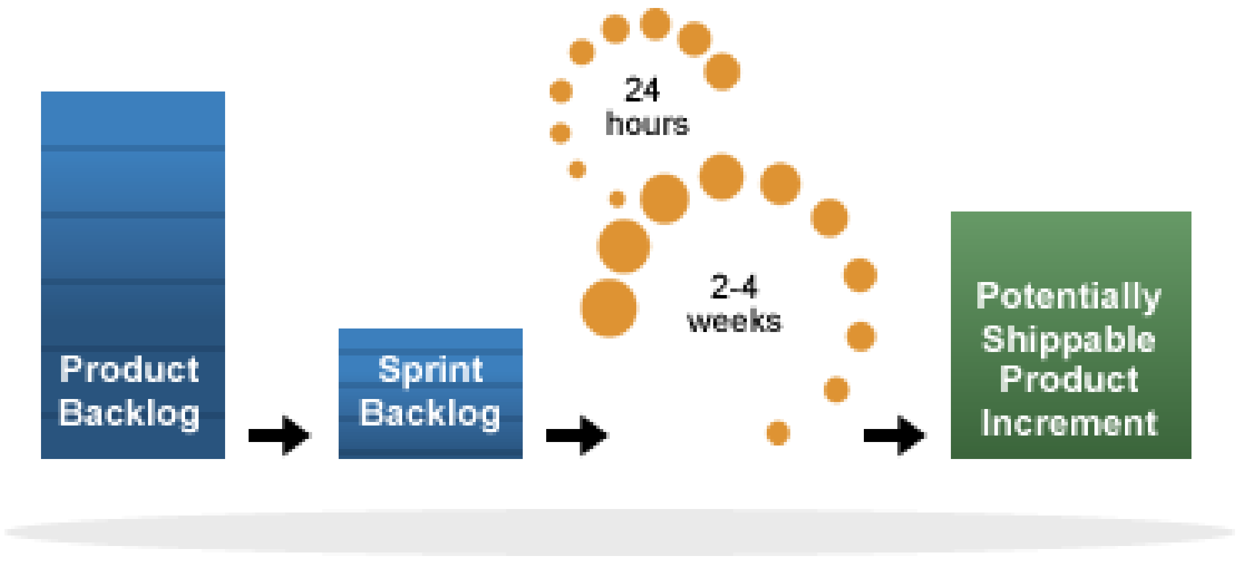


Figure 3: The Scrum Framework

a. Preparing

At the beginning of the project, Thien - the team leader gave the members a raw version of Product Backlog, in which contains requirements that the game should have.

b. The Sprints

After having the complete Product Backlog, the Sprint Planning Meeting is organized and the Backlog will be analyzed, prioritized and agreed by all members of the team. During the meeting the team also selected the Sprint Backlog for the following Sprints (as shown in the simplistic Product Backlog table below).

Sprint	Work
1	- Generate Random Developers - Generate Random Projects - Hire Developers - Fire Developers - Assign Developers to Projects - Remove Developers From Projects - Accept Projects - Cancel Projects - Buy + Use Items - Buy + Use Hardware - Display Current Week - Add Sound - Add image - Calculate Money - Sort game - Pause Game - Display Project Details

	- Display Developer Details - Save Game - Load Game
2	- Level Up Developer - Calculate Function Points - Unit Testing - Create Special Events - GUI Animation - Different Difficulty Levels - Project Income - Developer Salary

As scheduled, all Sprints in "DevFortress" lasted 4 weeks and the first Sprint began at the day after the Planning Meeting. Tai ensured that the Sprint Backlog would be updated throughout the Sprint by all members of the team. Every morning during the Sprint, the team had a quick stand-up meeting to report the working status.

There is a Review Meeting after the first Sprint. At this meeting, the game will be demonstrated the parts of code which we had finished in this Sprint. With the agreement of Dat, the User Interface is released The Product Backlog might be changed in this meeting regarding to the team leader. Retrospective Meeting was also organized by Hung for the team to recap what they had done in the Sprint.

After finishing the first Sprint, the Planning Meeting was organized again to prepare for the second Sprint. The project was continued as long as the team leader agreed that the code was fine.

VI. Changes and justifications

a. New features

- Discard Project: you can cancel a project after accepting it. However, it will lead to a penalty of 50% of the project price. In addition, if you cancel a project within the final 2 weeks of the project, you will be charged 75% of the project price.
- Fire Developer: if a developer works inefficiently, we can fire him. If you do so, you have to pay him that month salary even if it is not the end of the month.
- Monthly Report: the game will generate a report which will show details information about what you have done in that month. For example, the function points being produced and used.

We also add some events for the developers.

- EventThieves: the rate it might happen is 0.01. When it happens, the company will lose 20% of the current budget.
- EventIncreaseSkill: the rate it might happen is 0.05. One of the skills of the developer will increase by one level. That skill will be chosen randomly.

b. Changed features

- Change Skill Cost: Depends on the difficulty level, the Skill Cost will be different. The harder mode you play, the higher the cost will be.
- Produced Function Point Calculation Formula:
  Function Points = ((technicalLevel + (2 * designLevel) + (3 * analyLevel) + (technicalLevel * algoLevel) + (teamLevel * teamMembers)) / (11 - configLevel)) * (pizza && coffee && redbull ? 1 : 0)*(beer ? 0.5 : 1)
- Starting Budget: it will be changed due to the difficulty level. The hardest mode will provide you the least starting budget.

VII. Additional constraints

- This game is written by Java and it is compatible with several OSs like Windows or Linux.
- The game saves file locally, so it does not require any connection to an online database.

VIII. Technology stack and tools

We used several tools during the process of developing DevFortress game.

- Netbeans was used mainly for coding.
- Several libraries were used:
    - WebLookAndFeel and Synthetica were used for making GUI.
    - Jfreechart and JCommon were used for making charts.
- For the diagrams, we used StarUml.
- Because we could not be able to work together every day, GitHub was used for storing the project. Then we can upload our work and others can get it easily.