TDT4240

SOFTWARE ARCHITECTURE

SPRING 2012

GROUP A17
ANDROID SDK

Steinar HARAM
Emil Andreas MORK
Stian SØREBØ
Ole Jørgen RISHOFF

Architecture Document v1.0

PRIMARY FOCUS:
MAINTAINABILITY

SECONDARY FOCUS:
USABILITY

Contents

List of Tables

List of Figures

1 Introduction

According to the recommended practice of IEEE 1471, this document details architectural patterns and tactics based on all of the underlying drivers, stakeholders and their concerns. The 4+1 View model is a recommended practice for architecture description of software-intensive systems, and will be used to describe the system from the viewpoint of different stakeholders, such as end-users, developers and project managers.

2 Architectural drivers

Project Selection was based on being able to use the architectures that were used in the course. This is to maximize the learning outcomes and make the project feasible. We also want to create an architecture that can be developed without any big problems. On these criteria the needed business qualities are listed in the following sub-sections.

2.1 Quality in all parts of the architecture

To create a good architecture we want to make sure all parts of the documentation is well thought through, and that all parts are connected and understandable for all group members.

2.2 Not too complex architecture

The project lifetime is only 10 weeks. We must limit the functionality and possibilities of the project so that it becomes feasible.

2.3 Fast start up

The faster we get the architecture done, the faster we can start programming. We will try to be one week a head of the plan. This will make our schedule resistant for problems that might arise.

3 Stakeholders

Stakeholder	**Concern**
Developer	Buildability: The game needs to be finished within a short period or time. Modifiability: The game should be easy to extend and change current features. Testability: It should be easy to test whether the game is working as intended or not. Grade: The project evaluation criteria will be prioritized to achieve the best grade possible.
Player	Usability: It should be easy to both use and learn by new players. Playability: The game should be interesting and fun to play. Easy setup: The knowledge and time needed to install the game should be trivial.
Course staff	Reviewability: The delivered code should be readable, and the setup should be easy.
ATAM evaluator	Reviewability: There should be good architectural documentation that consist with the code.
Android Market	The final application should follow the guidelines for publishing applications on Android Market.

Table 1: Stakeholders

4 Architectural viewpoint selection

The 4+1 View Model consist of four different views: Logical View, Development View, process View and physical View. Our game will be a 2-player network game, but since we are going to use a third party library for the network communication we will not use the Physical View.

4.1 Logical view

Basis	We will put the system into perspective and get an overview of the software architecture. Standard procedure is to divide the program into different object-oriented models and find their relationship.
Stakeholders	Developers, course teachers
Description	Standard Class diagram with package notation

Table 2: Logical view

4.2 Development view

Basis	Used to get a perspective view of the different main blocks of the systems.
Stakeholders	Course teachers, Developers
Description	Layer diagram

Table 3: Development view

4.3 Process view

Basis	We use the process view to get an overview of the process flow. This can help the developers to design and understand the main logic and structure of the game.
Stakeholders	Developers, Course teachers
Description	Activity/Sequence diagram

Table 4: Process view

5 Architectural tactics

5.1 Modifiability

5.1.1 Prevention of Ripple Effect

The code should contain as few dependencies as possible in order to avoid ripple effect when making changes to the code.

5.1.2 Localize Changes

As our quality requirements highlight in the Requirements Document [1], we have anticipated several possible extensions to our game application. We have taken this into account, and will build our architecture with this in mind.

5.2 Usability

5.2.1 User feedback between implementation iterations

To optimize the usability of our user interface, we will get feedback from outside users during the project.

5.2.2 Follow the Android user interface guidelines

These design principles were developed by and for the Android User Experience Team to keep users' best interests in mind. We will consider them as we apply your own creativity and design thinking.

5.3 Testability

5.3.1 Testing third party multiplayer connection

The application uses a third party multiplayer feature. This makes connection testing easy, as it is detached from the rest of the application.

6 Patterns

6.1 Architectural patterns

6.1.1 MVC

Model-View-Controller (MVC) is an architectural pattern that divides interactions between users and applications into three roles: the Model (business logic), the View (user interface), and the Controller (user input). This separation makes is possible to develop, test and maintain each role independently. Model-View-Controller pattern will be used when implementing *Nine Men's Morris*, where we will isolate the domain logic from the user interface. This will separate our program into distinct features, which will be easier to develop, test, and maintain.

In Android the Activities will be our View-and-Controller, so as the *View* it will need need to implement the *Observer* interface. As the *Controller*, it will need to implement the *OnClickListener* interface. The Activity's methods are going to access the *Model.*

6.2 Design pattern

6.2.1 States

The state pattern will be implemented in order to ensure high modifiability, and to be able to smoothly change the state of the game at runtime. The different states of the game is shown in the logical view in section 7.1.

6.2.2 Singleton

To avoid unnecessary instantiation of objects, we will implement the singleton pattern where we see fit. This will benefit the developer that only needs to deal with one instance of the object in question.

7 Views

7.1 Logical view

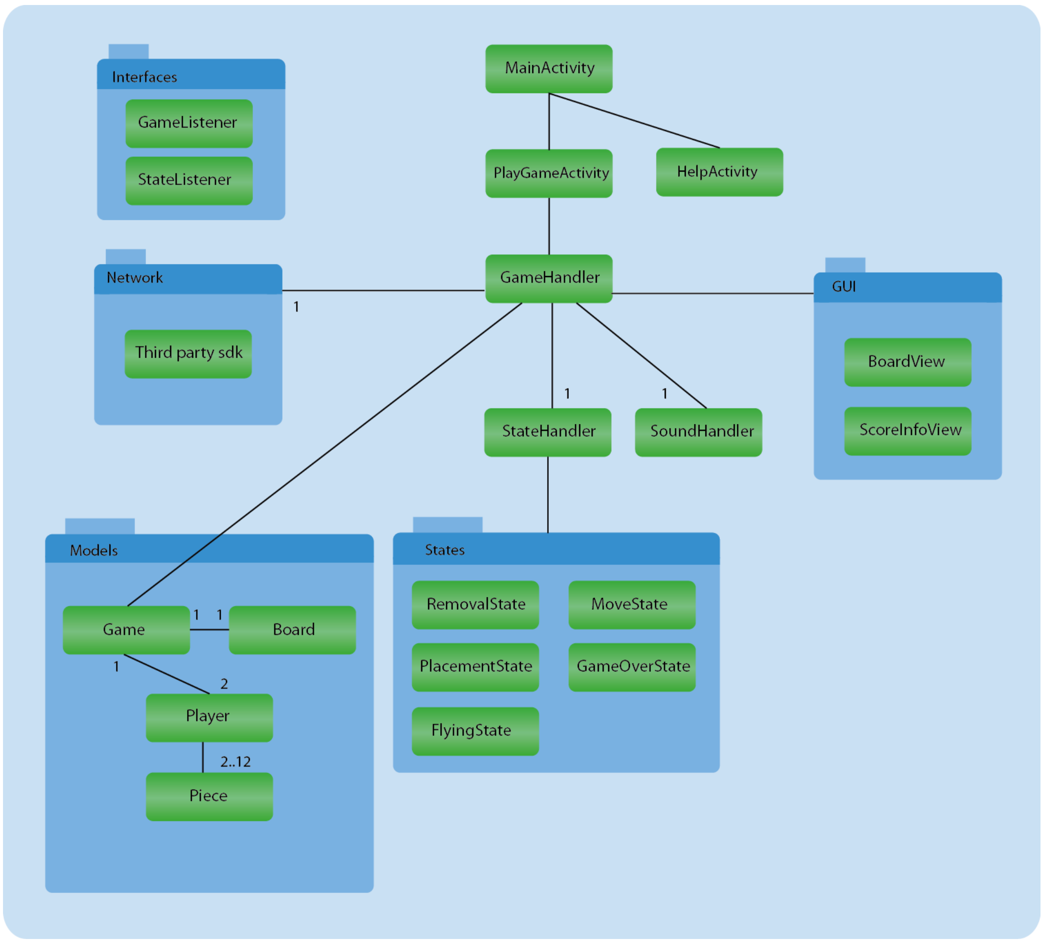

Figure 1: Logical view

The diagram follows the 4+1 logic view notation suggested by the Kructhen article [2]. The class diagram shows the structure of a system by showing the system's classes and the relationships among them. Aggregation and inheritance is also displayed.

7.2 Development view

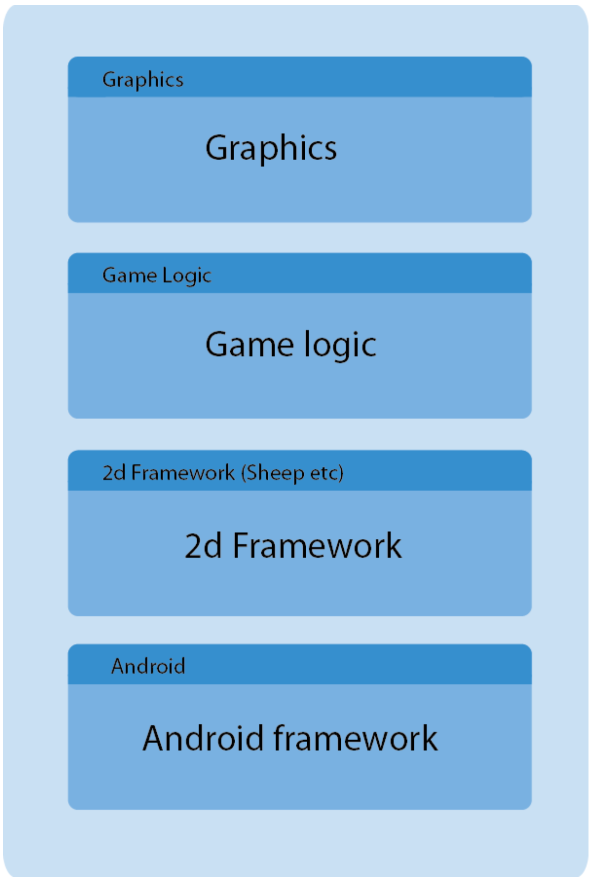

Figure 2: Development view

The game is divided into several layers. The android framework, a 2D framework, the game logic, and graphics. The diagram show the different parts of the system.

7.3 Process view

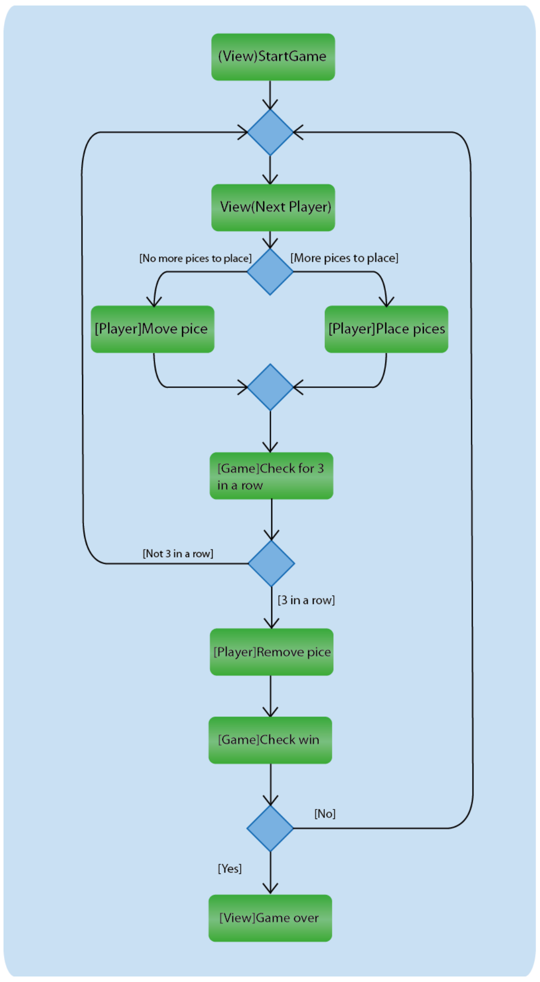

Figure 3: Process view

The game is a turn based game with two players. In the activity diagram the progress of a game is described. When a game is started one of the players starts placing one of its pieces. The game checks if the player get three in a row. In that case the player can remove one of the opponents pieces. If the opponents have less then three pieces left, the other player have won. The diagram shows how a game round will pan out.

8 Consistency among views

We are using three views which have consistency with each other. The point here is to make the views understandable for different stakeholders, and help them easily see the different aspects of the program.
The different states and their relations are described in the Process view. In the Logic view we illustrates how the different parts of the program is implemented. And the development view shows an overview over the different tools and framework

9 Architectural rationale

We have chosen to build our architecture with implementation of the MVC, State, and Singleton pattern. The MVC pattern will help us cleanly separate domain logic from the user interface, which conforms with both modifiability and testability. The state pattern allows for extension of the game in a convenient matter, conforming with our main focus. The singleton pattern is used to prevent unnecessary instantiation of objects central to the application.

Usability is exstremely important in a gaming environment, and we have therefore chosen to have this as our second area of focus. The end user will have a lot to say in our final design, and we will strive to satisfy this user group in a best possible manner.strive to satisfy the end users in a best possible manner.

10 Issues

We have had some difficulty finding a multiplayer framework that suits our exact needs, but we do not see this as something that will hinder our progress significantly.

11 Changes

Date	**Change**
27.02.12	First version of the document

Table 5: Document changes

References

[1] S. Mork, Haram and Rishoff, Requirement Document.

[2] P. Krutchen, The 4+1 View Model of Software Architecture, `http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf`, 1995.