

# Dominion

Analysis, Design and Software Architecture

Jakob Melnyk, jmel@itu.dk  
Christian Jensen, chrj@itu.dk  
Frederik Lysgaard, frly@itu.dk

December 14th, 2011

## Abstract

This project is about a virtual representation of the card game Dominion in C#. Dominion is a turn-based, deck-building game, where the objective is to gather more points than the other players. The game is played by 2 - 4 players.

# Contents

<b>1</b>	<b>Requirements</b>	<b>4</b>
1.1	Mandatory . . . . .	4
1.2	Secondary . . . . .	4
<b>2</b>	<b>Overview</b>	<b>5</b>
<b>3</b>	<b>Dictionary</b>	<b>6</b>
3.1	General terms . . . . .	6
3.2	In-Game terms . . . . .	7
<b>4</b>	<b>Example</b>	<b>9</b>
4.1	Starting the game,hotseat or LAN. . . . .	9
4.1.1	LAN . . . . .	9
4.1.2	Hot-Seat . . . . .	9
4.2	Getting started, what is Dominion really about? . . . . .	9
4.3	T . . . . .	10
4.4	e . . . . .	10
<b>5</b>	<b>Revision History</b>	<b>11</b>
<b>6</b>	<b>Milestones</b>	<b>14</b>
6.1	System analysis . . . . .	14
6.2	System design . . . . .	15
6.2.1	General . . . . .	15
6.2.2	GUI . . . . .	15
6.2.3	Client & Server and Control (Server and start-up parts) . . . . .	15
6.2.4	Gamestate and Control (Game Logic) . . . . .	16
6.2.5	BON specification . . . . .	16
6.3	System production . . . . .	17

6.3.1	General . . . . .	17
6.3.2	GUI . . . . .	17
6.3.3	Server and Control (Server and start-up parts) . . . . .	17
6.3.4	Gamestate and Control (Game Logic) . . . . .	19
<b>7</b>	<b>BON-specification</b>	<b>20</b>

# 1 Requirements

## 1.1 Mandatory

Must be able to play a full game of Dominion

- Must support 2 players in Hot-Seat configuration
- At least 10 Kingdom cards must work
- The game must be playable in a Picture-based GUI

## 1.2 Secondary

High priority

- Be able to play the game with 3 or more players
- Be able to use at least 20 Kingdom cards
- Be able to select Game Mode
  - Be able to play 'First Game' Card-set
  - Be able to play with 10 randomly select Kingdom cards
- Be able to see all Available Kingdom cards without scrolling

Medium priority

- Be able to view a Tooltip when mousing over any Card in the game
- Be able to play the game over LAN
- Be able to use all Kingdom cards (from the original version of the game)
- Be able to play all the Card-sets defined in the original rules

Low priority

- Be able to Draft Kingdom cards
- Be able to play the game over the Internet
- Be able to select different screensizes
- Be able to play in fullscreen

- Be able to create a User, that is saved across multiple games, with the following information:
  - Statistics
  - Options (if any)
  - Achievements (if implemented)
- Be able to support Extensions of the basic game
- Implement Achievements for funny and/or hard accomplishments

## 2 Overview

This project is about our virtual representation of the card game Dominion. Dominion is a turn-based, deck-building game. The objective of the game is to use Action cards to improve your chances or damage the opponent players and using Treasure cards to buy more powerful Action/Treasure/Victory cards to gain the upper hand.

We are planning on using a Model-View-Controller architecture. We want to separate our GUI from both the game rules and the state of the game via the controller. In essence we are likely to have a somewhat static model of the rules and a more dynamic and changing model of the state of the current game.

Frederik Lysgaard is the guy responsible for the design of our graphical interface. He is also the best Dominion player in our group. Because of this, he knows a lot of the usual strategies and is our general "go-to" guy when it comes to the tactics of the game.

Christian Jensen is responsible for implementing the way the different cards interact with the state of the game when used. Christian is also the guy who will be looking into the networking portion of the project if/when it becomes relevant.

Jakob Melnyk is responsible for modeling the state of the game and the communication between the GUI and the model (in our model-view-controller architecture). Jakob Melnyk is also the "version-control-guy", the person with the final word in discussions and the general log-keeper for the group.

## 3 Dictionary

### 3.1 General terms

This section describes the general "out-of-game" terms.

**Achievements** An achievement is token rewarded for funny and/or hard accomplishments within the game.

**Card-set** A card-set is 10 different Kingdom cards. Card-sets are used to create a different play experience every time you play.

**Dominion** The card-game we are making a virtual representation of. A link to the full rules can be found at Rio Grande Games [2].

**Draft** Drafting is done by player 1 selecting one Kingdom card to be used in the game, then player 2 selects a Kingdom card, player 3 selects a Kingdom card, player 4 selects a Kingdom card, then back to player 1. This cycle repeats until a set number of Kingdom cards have been selected.

**Extensions** Expansion packs add additional types of cards to the pool of cards.

**Game Mode** There are different possible game modes: draft, random card selection and predefined card-sets. These are selected before the game starts.

**Hot-Seat** Hot-Seat is the act of having 2 or more players play on the same computer. The active player "sits" in the hot-seat while playing, then passing the spot to the next player when his turn ends.

**Message Type** Messages of different types can be passed around in our server-client network.

**Model-View-Control** Often abbreviated MVC, Model-View-Control is often used to separate something "showing" data and the actual representation of the data on the disk. Control is usually the middle-link that takes care of the communication between the two.

**Picture-based GUI** A pictured-based GUI is a visual representation of the state of the game. The different cards are shown as pictures in the GUI.

**Server-Client** In a client-server design, the clients communicate with the server and the server then relays the information it was given by the client to the other clients.

**Statistics** Statistics such as number of games played, numbers of games won/lost, and other similar data about gameplay.

**Tooltip** A box with text describing something in the GUI in detail.

**User** A user is an entity storing statistics and achievements over the course of different games.

## 3.2 In-Game terms

This section describes the types of cards, supply and other "in-game" terms.

**Available** Available Cards are the Cards that can be bought from the Supply.

**Action Phase** In an action phase, a player have one Action, which he or she may use to play an Action Card. Playing an action card this way always costs one Action. Cards played may allow a player to receive additional actions. The Action Phase ends when a player has no more Actions left or chooses not to use his or her remaining Actions.

**Buy Phase** When a player's Action Phase ends, the Buy Phase begins. In this, the player receives a "Coin" amount, which is the combined value of all Treasure Cards in his or her hand and any Action Cards, that add "Coins". The player can then use a Buy to buy any Card they want from the Supply. Played Action Cards can allow more Buys. Bought Cards are added to the Discard Stack. After the Buy Phase, the Clean-Up Phase begins

**Card** A Card is the basic playing unit in Dominion. Everything you 'own' is represented by a Card in your deck. Cards are primarily added to the deck through the Buy phase. Each Card has a value, which represents what it costs to get during the Buy Phase.

**Curse Card** A Curse Card is a special type of Victory Card, which gives a negative amount of Victory Points. While these cards can technically be bought by any player and added to his or her deck, they are usually given to other players by using Attack Cards against them.

**Kingdom Card** Kingdom Cards are what make each game of Dominion unique. With one exception all Cards here are Action Cards (one is a special Victory Card) and there are no Action Cards which are not Kingdom Cards. Each game requires selecting 10 of the 25 Kingdom Cards to use.

**Action Card** An Action Card is used during the Action Phase.

**Attack Card** An Attack Card is a type of Card which affects other players negatively. All Attack Cards are Action Cards and the "Attack" activates when the Card is used as an Action.]

**Action-Reaction Card** A Reaction card is used to respond to an Attack by another player. When an Attack Card is used against a player, that player may reveal a Reaction Card from his or her hand and do what the Reaction allows. Only one Reaction Card is in this game, 'Moat', which allows the player to negate the attack used against them.

**Kingdom Victory Card** A Kingdom Victory Card is a card that does generally not behave like usual Victory Card, but instead have special effects granting the player Victory Points.

**Treasure Card** A Treasure Card adds a number of "Coins" to spend in the Buy Phase. Note that a Treasure Cards value (the price to buy it) are usually different from what they cost to buy.

**Victory Card** A Victory Card gives a number of Victory Points at the end of the game. The player with the most Victory Points win the game.

**Clean-up Phase** The Clean-up Phase consists of putting all bought Cards, played Cards and Cards remaining in the Hand into the Discard Stack.

**Deck** A players Deck is his or her representation in the game. It consists of all the Cards that player started with and have bought during the game. A player's Draw Stack, Discard Stack and Hand is that player's Deck.

**Discard Stack** This contains previously played cards and any newly bought cards.

**Draw Stack** This contains face-down Cards for a player to draw. When there are no more cards available for a player to draw, the Discard Stack is shuffled and used as a new Draw Stack. Each player have their own Draw Stack and Discard Stack.

**Hand** The Hand represents a players current options in the following turn. These are drawn at the start of the game and each player draws a new hand after a turn has finished. When drawing a new hand, it always consists of 5 Cards.

**Supply** The Supply consists of 10 types of Kingdom Cards, 3 types of Treasure Cards, 3 types of Victory Cards and Curse Cards.

**Round** A game of Dominion consists of a number of rounds. Each Round is divided in to Turns, one for each player.

**Trash Stack** Sometimes a Card calls for itself or some other card to be Trashed. This means that it should be completely removed from the game and the Trashed Card is put on to the Trash Stack. All players share the Trash Stack.

**Turn** The player usually take turns in clockwise order. A players next Turn will be in the following Round.



## 4 Example

**Frederik Roden Lysgaard** This will be an example of our project, which is a graphical representation of the cardgame "Dominion", published by Rio Grande Games. The walk-through will be built up around certain screenshots and will cover the following points:

- Starting the game, hotseat or LAN.
- Getting started, what is Dominion really about?
- The user interface.
- End of game.

### 4.1 Starting the game, hotseat or LAN.

#### 4.1.1 LAN

When starting the application you will be presented with a console window asking you to take the role of either Client or Server. `pic(clientinput)` often one of the players will choose server and will then be able to give his fellow players who chose client, an IP to connect to. `pic(startserver)` when the appropriate amount of players has joined the server (usually 3-4) then the person with the server program running will call `startgame` (command is `<STGM>`) and the game will then start. `pic(startgame)` `pic(startscreen)`

#### 4.1.2 Hot-Seat

This is done almost identically to the procedure for a LAN game the only exception is, that instead of letting other computers create a client, you just run multiple instances of the application on your computer like so: `pic(serverclient)` and after that it's just that same as with LAN games.

### 4.2 Getting started, what is Dominion really about?

Dominion is a deckbuilding game which means, that the object of the game is to build yourself a deck which will give you the best hands, and thereby giving you the edge in getting the most victory points which in the end determines who wins. I already introduced some of the game-specific words and I will now show where they are placed on the playing board and what their responsibility is:

## 4.3 T

he user interface. `pic (actioncard)`

- hand The hand is where you see what you have drawn each turn. In Dominion there's three kind of cards: Treasure, Victory or Action all three kinds can be drawn into this field. If you click on a Action card while it's placed in hand and you got actions left then the card will be moved from the hand to the actionzone.
- actionzone The actionzone is where the actioncards that is played from the hand is shown. Only actioncards can be drawn in this zone. When a turn ends the actionzone will be cleared and the actioncards will all be moved to the discard.
- discard the discard zone is where the cards go when they are not in use anymore, you can't click on cards while they are in the discardzone, while in the discardzone the cards can only wait to be shuffled into the deck again.
- deck the deck is where the cards is held until they are drawn at new.
- supply the supplyzone that are drawn to right side of the GUI is crucial to the game this is where you can buy your kingdom cards and there by increase your decks size and strength, as we can see, we draw both the seven static victory/treasure cards and 10 extra kingdomcards, which also was one of our mandatory requirements.

## 4.4 e

nd of game. As stated by the Dominion game rules the game ends when either the province victory pile is empty or three kingdomcard piles are empty. When this happens we check which player has earned the titel of winer. If you are indeed the winner then you will be greated with a lovely congratulations message printed across the screen. `pic(youarewinner)` but if you loose you will be meet with disgust. `pic(youloose)` So to summarize our digital representation of Dominion let's you play with 10 preset kingdomcards in graphical interface with your friends either over hotseat or local area network. Which by total coincidence also is our mandatory requirements.

## 5 Revision History

Some of our commits to our Github (<https://github.com/esfdk/BDSADominion>) with commit ID. The commits are sorted by time of commit, descending. Full commit log can be found at: <https://github.com/esfdk/BDSADominion/commits>

e92c7c5381f65ca8f3deef9458922373c4da81de : Put BON files into Hand-in.

940473cec8b897f58774415fa99eb6fd50771b07 : Merged dotCover file added. System design and production of network added.

---

Code freeze

9d86882ca255502d2cd07bdfbc13e84b13d50fb8 : Lots of work on hand-in getting done.

c2dcc6786210ecf5ac74a8d34bcf07e72cf8360c : pex snapshot

a397b2b79291e8349e106ef1773406e3c3a57e55 : made a dotcover solitaire run

cae05ae98caefbeaf2cf31fa4eb9473507486ce1 : contract

00782159d83baf89bf3a41acea7f87d6484e263f : network design done.

28cb2b5c8f343793662fa0cec0d8dae0e8476564 : Fixed bug in shuffling of cards. Not as random as it was before.

234ca16500ee52cbf8f37dff117de83c07febebe : informal is now legit

e22f402855a48cc2ce93ce651305bb801e121d75 : Key and control update

fe7e9ba83b41594a5ef91560af0fb505a55da627 : Some pex tests, better shuffle method, fixed adventurer and more.

3074885da9f22646bac0becbfe8bc45ae975723a : Made some changes to GUI it looks good now

e083766fe15b127f0a9ae03528e1eea956a6702e : dotCover solo run added

1951adb1e07b4d9735847e0bf3e622464380aa2f : Trying to get these weird bugs out of the game.

7c53affdf9aba583efc71dedab9476fe1441eeea : Done with some of the docs for the GUI

1173a01763032deb6b033d01471039be84a61f0f : Wrote some of system design and started on system production

a35c4939236770661fb751f378be4c7e5259d0fe : Formal bon 99% done for Melnyk.

97ae13e744b2283f0a0aeca954b357d0e2c04 : Minor fix in Control to possibly prevent crashes. Also added some more Contracts to the formal BON.

aa1c0af808bc536cc50e5e988e185834ca34bd13 : GUI start update working now

9d4d9b2bcf0b8acda2544fd3c93ad926122237c4 : added the listeners

44f6bb367f5b56014069c2bfd8b1e8fd1a61da7c : Made files to be used in the hand-in.

911af507af6ec9e64a36d470280d1b4d7e9707d7 : Network starts the game

b7210739f2ed0b37875ab7d4a858559602e029b2 : Server PreGame messaging working. Still lots of WriteLine.

9a3c7bfc2334e04a0709a09eeb0d1b4a532327a7 : Much of Melnyk's part of Control is done! Mainly need communication with server!

3417bf70282d5b3bdfd39c8855f258eeb05e57a2 : Network seems to be working alright.

c710a8d4a93dfdc6623be4022bcb8da2a0259d4d : GUIInterface done. Network testing

bb1b5dbabccfc6c7d3c1d25ffa97fcf5e8d37b64 : Starting tests on network interface

537f2f25db57b263d63cac1ab298a69de5e80bf3 : GUI-Interface almost done, only need endPhase button in gamestate.

9f2a6f20d46e0037a2dfa2d917b3030b720551c3 : GUI Interface getting up and running. Console added :D

a883a6fd4d552fe71cb3aae641a7ec0af4037e06 : GUI now draws what needs to be drawn

7c368444740ed2bc8d49027e7a7d666674600ec2 : Finished gamestate and player for now

24454e804bde99435b2022244e6307d89a9c323a : GUI now updated to work with Card-Name

1ddb9247acd2f32b0e979f7dd561bd98edf4d570: Alfa GUI is done. LOCK AND LOAD!

5c28c48e556fea810b4dec2e3db4108f6c455ae4 : Diary and some gamestate stuff done.

857d25376348dd601579e20899eb76efc074a30d : Networking added. More or less direct copy of work done outside of project. Interfacing with Control added, but not completed

bec4f5f40ae4aca6189ab41078fbd9c0475ca8e5 : Started coding gamestate

523bd81b3a6e0e20185ff5e2b3aa5fce13ccbc25 : Arranged classes into folders for better overview.

ff7248e0612f9f2e53525fde41de536b22d624b7 : Started creating gamestate in code

5e858aff8cab0331b3a504f4d4280a4dbf774cc7 : Did lots of BON

1d7eb386c7839a9c97ac1b8efb286b12fdaf7d97 : Added all the necessary classes for the GUI part of the project

6af320925f540af6842e9b6156355738d3cc94b1 : Did a lot of informal BON

25ff80260b970ec14e36728249e3f9cb324052b0 : Did some BON and added some diary entries

e0ce1ea7ec8bd8345bfb6f66c77c3d908d40af03 : Put in BON files

86029d88fde9a403f777855e433cd3fd8775ff7e : Added diary files

533e909a8d36dbffb1c6953cf99639a5baa90fbc : Added GUI classes

905a2d3a02218dec666bff65c4738306e9e7877c : Starting project

## 6 Milestones

### 6.1 System analysis

## 6.2 System design

### 6.2.1 General

We decided to use an MVC pattern for our overall structure, because we felt we could separate the model (the GameState), the View (GUI) and Control (Game Logic and more). We also use a Client-Server architecture to do the network communication.

### 6.2.2 GUI

**Frederik Lysgaard** In terms of system design, there isn't really much to say to the GUI part since none of us knew the XNA framework, and therefore didn't have any knowledge of the limitations and benefits. This led to a very ad hoc way of designing the GUI system, for the first draft I used my extensive knowledge of the game to try and get an overview of the components needed to properly display and play a game of Dominion. This resulted in the basic design for the GUI, but since I was still a beginner to XNA, this led to a design with almost all the right classes but with some strange inheritance, all in all it felt weird which made me study XNA some more, and after having used some days getting familiar with the framework I came up with the second and final draft.

Even though this was the final draft, it was far from perfect, I would for example if I had, had more time with XNA before the project suggested some inheritance between the zones since they all are made from the same sprite "template". So to summarize there wasn't really any general battle plan for the GUI system design at the start of the project, which was a challenge, that taught me one thing, if you know you're going to work in a new framework, learn it beforehand.

### 6.2.3 Client & Server and Control (Server and start-up parts)

**Christian Jensen** When we set out to make Dominion, we knew that network should not be a mandatory requirement, yet here we are with it. There are several reasons for this, such as that even to be moderately able to play the game, multiple computers should be involved. However, on a more practical level, I lacked an area of responsibility after we decided to couple the Control and GameState tighter, and network seemed an obvious idea.

I had one problem however: I had little to no idea about how to build a network system that would be appropriate for the project (nor did any of my teammates), both in terms of C# and networks in general. This is the primary reason why I could not sit down and design a network off my head. I needed to study the options first.

I did what any respectable coder would do; I searched the internet to see if somebody already had coded a solution. I came by several solutions but they were either not doing what we needed or was too complex. An answer presented itself. I knew of a guy who is in a group doing a separate project and who is a wiz at networks in general

(Simon Henriksen, shen@itu.dk) and the network part for their project (Descent) was quite similar to what we needed. We had several inter-group meetings during the project period with his group and a third (Magic the Gathering) and I asked him to help me understand several concept related to networks and how C# could use them.

He often referenced to his own implementation and just like using an Internet example to provide the basis for an implementation, he graciously allowed me to use his implementation as the basis for mine. I am not peticularly proud of this point, but my options was either use some structure I didn't really understand or use this, which I will claim to have grasped quite well.

#### **6.2.4 Gamestate and Control (Game Logic)**

**Jakob Melnyk**

#### **6.2.5 BON specification**

We have included all of our BON in section 7 on page 20.



## 6.3 System production

### 6.3.1 General

Our split into the different, very separate parts of the code, made it somewhat cumbersome to combine at the end, but once it actually combined, it was quite an easy ride home in terms of getting the game to play. The different parts of the architecture should be quite replaceable, especially considering the `GUIInterface` and `NetworkingInterface` concepts and the way `Gamestate` works.

### 6.3.2 GUI

**Frederik Lysgaard** The production of the GUI can be split into three parts:

- The initial idea.
- The attempt to write it.
- And at last the rewrite of it all.

So let's start at the beginning. The initial idea of how to produce the gui was that all drawn classes should inherit from a super `Sprite` class but as I began coding I realized that the idea wouldn't be so optimal, since we had different objects with different positions which at that point, in my XNA training, seemed to make it all very hard to draw, at least with different positions.

So after realizing that my first attempt of code was not going to work, I set to rewriting what I already had and try and reform it with my new knowledge of XNA. I then ended up with what is our end GUI which consist of a lot of zones where you can either draw buttons or cards sprites to, this seemed like a extremely easy straight forward solution, even though if I had had more time, I would have loved to code in some inheritance, especially a super zoneclass that would act as template for the other zoneclasses.

### 6.3.3 Server and Control (Server and start-up parts)

**Christian Jensen** We had talked about networks and it was clear that we didn't want to communicate directly with clients and server. An interface was needed. `Dominion` has several cards that require a player to respond to how they want to react to it when another player plays it. This means that we needed to ensure that the other players has a chance to make a choice if such a card is played. The way that this works is through the `TurnMessage` method, through which all communication takes place.

In `TurnMessage` a arbitrary string is passed to the `NetworkInterface`. The only requirement is that it does not contain `'—'`, `'ı'` and `'ç'`, as these characters are used by the system. This is then wrapped in 2 pieces of information: a `MessageType`, which tells the

server and the other clients what kind of message it is, and also added, is an piece of text that signifies the end of the message.

This is then sent to the server. The server communicates with the clients through a number of Connection objects, one for each client. These also contains the Socket objects on the server side. The Socket listens to incoming messages thanks to the BeginRecieve method, which then calls BeginRecieveCallback when a message is received. This method and the asyncResult it takes as a parameter, along with BeginRecieve, is the basis of the entire network.

The message is then passed to the server itself through an event. This ensures that the server can take any messages it receives at the "same" time. The message passed to the server no longer contains the End of File textpiece, but still contains the message type. The Server can then use the message type to determine how to act with it. In most cases, the server will simply forward the message.

The server keep all its Connections (which basically is its clients) in a Dictionary, with the Id of the player as the key. As keys are created incrementally as clients join before the game, 1 is defined by us as the Host, since the Host joins almost instantly after creating the server. The server can then use this Dictionary to forward the message, by excluding the Connection with the Id matching the Connection that the message was received from.

The message gets reattached with its message type and also joining it is the Id of the player that sent the message. The Client recieves it in almost the same way that the Connection does; through a BeginRecieve and a BeginRecieveCallback. The Client then passes the message to GuiInterface, after removing the End of File text.

Then we are back at the GuiInterface, this is where things gets interesting. Depending on the message type, the GuiInterface reacts differently. System messages are passed on through the event MessageReceived. Action messages are also passed on, but they also flings a message of the third type back through the system, the Response. Response messages are stored in an array, where they await something to pick them up.

The Response message is how I have planned to implement cards that require a reaction from another player. The Idea is that on all normal action cards (i.e. cards that doesn't require another player to actively make a choice) a 'message received' Response, sent from every other player and this is then the 'insurance' that all players have received the message by the TurnMessage waiting for replys from all other players. These 'message received' returns are sent to all players, not just the active, so that everyone can be sure that everyone is still in the game. However, on a card that require a real reply, it would instead send a WaitResponse type message, which would tell the sender that they should wait, because a message from that player was coming later.

Almost the entire system for this is in the code, apart from the infrastructure to WaitResponse. We have however chosen not to enable any of it in the build. We found that for one, it causes bugs in rare cases. Network is a fiddly thing to test and bugs happened, one reason being that messages were received too quickly after one another, due to the Threads that lie behind the listening structure. Another reason to disable it is that with the current structure we have no cards which require replys from any player that is not

currently active. Basically we prioritized our mandatory requirement, that the game should be able to be played, rather than implementing a perhaps overly fancy feature.

PreGame: The server needs to set up the group of people playing. This is accomplished through these steps: first a person creates a server through the console at the beginning of the game. This command creates a server and a client object, the last of which instantly connects to the first. He is then shown his IP, which other people can connect to, by creating a client instead of a server and typing the IP address of the host. To begin the game, a special command is required. All players can type messages, which are sent with PreGameMessage, which uses System message type instead of action. They are sent to all players and shown in the Console. The server is hardcoded to wait for the command 'STGM;' from client 1 (the host). This will cause the server to send a system message to all players, who are in turn hardcoded to wait for it, because it contains 2 key pieces of information: The total number of players and the id of the player. After this is received, the game will begin.

Obviously a console start-menu is not optimal, but the primary reason for not having this as part of the GUI is because the GUI lagged behind and we needed to fulfill our mandatory requirement: The game must work.

While the grand design might not have been there from the start, thought has gone into how the network should run, as this text should show.

#### **6.3.4 Gamestate and Control (Game Logic)**

**Jakob Melnyk**

## 7 BON-specification

```
1  system_chart BDSADominion
2  indexing
3      author: "Frederik Lysgaard (frly@itu.dk), Christian 'Troy' Jensen, Jakob Melnyk (jmel@itu.dk)"
4      supervisor: "Joe Kiniry";
5      course: "BDSA-E2011";
6      created: "28th November 2011";
7      lastModified: "14th December 2011";
8  explanation
9      "System chart for the BDSADominion project in the Analysis, Design and Software Architecture"
10 cluster DOMINION_SYSTEM
11     description "The Dominion game system."
12 end
13
14 cluster_chart DOMINION_SYSTEM
15 class CONTROL
16     description "The man in the middle between the three parts of the architecture. Contains gam
17 cluster GUI
18     description "Used to display the current state of the model and to interact with the user."
19 cluster GAMESTATE_CLUSTER
20     description "The 'model' of the project. Remembers information about most of the states and
21 cluster NETWORK_CLUSTER
22     description "Communicates between different instances of the application across LAN."
23 end

1  cluster_chart GAMESTATE_CLUSTER
2      indexing
3          author: "Jakob Melnyk (jmel@itu.dk)";
4      explanation "The classes making up the 'dynamic' state of the game. This includes decks, han
5      class GAMESTATE
6          description "The overall 'state' of the current game."
7      class PLAYER
8          description "Represents a player and everything a player owns."
9      class ZONE
10         description "List of zones that can be targets of events."
11     cluster CARD_CLUSTER
12         description "The different kinds of cards."
13 end
14
15 class_chart GAMESTATE
16     indexing
17         author: "Jakob Melnyk (jmel@itu.dk)";
18     explanation "Keeps track of the players and everything the players share, such as the trash
19     query
20         "May I have a new gamestate with this set-up?",
21         "Who is the active player?",
22         "How many players are in the game?",
23         "Is the active player in the Action Phase?",
24         "Is the active player in the Buy Phase?",
25         "What cards are in the trash pile?",
26         "What players are in the game?",
27         "What does the supply look like?",
28         "Number of actions left?",
```

```

29         "Number of buys left?",
30         "Number of coins left?",
31         "How many points does each player have?",
32     command
33         "Make this player the active player!",
34         "Begin Action Phase",
35         "End Action Phase",
36         "Begin Buy Phase",
37         "End Buy Phase",
38         "Do Clean-up phase",
39         "Increase the amount of actions the active player has by this much!",
40         "Increase the amount of buys the active player has by this much!",
41         "Increase the amount of coins the active player has by this much!",
42         "That player gains this card type in this zone from the supply.",
43     constraint
44         "Can have 2, 3 OR 4 players.",
45         "Cannot begin Action Phase while in Action Phase or Buy Phase.",
46         "Cannot end Action Phase while not in Action Phase.",
47         "Cannot begin Buy Phase while in Action Phase or Buy Phase.",
48         "Cannot end Buy Phase while not in Buy Phase.",
49         "The active player cannot be made the active player."
50 end
51
52 class_chart PLAYER
53     indexing
54         author: "Jakob Melnyk (jmel@itu.dk)";
55     explanation "Each player is represented by a player object that keeps track of their decks,
56     query
57         "May I have a new Player?",
58         "What cards do you have?",
59         "How many cards do you have in your deck?",
60         "How many cards do you have in your discard pile?",
61         "What card is on top of your discard pile?",
62         "What card is on top of your deck?",
63         "What cards do you have in your hand?",
64         "What number are you?",
65         "What cards have you played?",
66         "What have you put in your temporary zone?",
67     command
68         "Move this card from that zone to the temporary zone!",
69         "Move this card from the temporary zone to that zone!",
70         "Move this card from the hand to the temporary zone!",
71         "Add this card to that zone!",
72         "Remove this card from that zone!",
73         "Draw a card!",
74         "Draw this many cards!",
75     constraint
76         "A player cannot have a card in his deck, discard pile, hand, or 'played field' that
77 end
78
79 class_chart ZONE
80     indexing
81         author: "Jakob Melnyk (jmel@itu.dk)";
82     explanation "Represents the values used to refer to the zones in the player class and gamest
83     query

```

```

84         "May I have the value 'v'?",
85     constraint
86         "The values allowed for this class are exactly one of 'DECK', 'DISCARD', 'HAND', 'SU
87 end

1  cluster_chart CARD_CLUSTER
2      indexing
3          author: "Jakob Melnyk (jmel@itu.dk)";
4      explanation "Cluster showing how card system works."
5      class CARD
6          description "A card."
7      class CARD_NAME
8          description "The names of all the cards."
9      class CARD_FACTORY
10         description "Produces cards."
11     cluster CARD_TYPES
12         description "The different meta-types of cards."
13     cluster CARDS
14         description "Contains all the cards from the game."
15 end
16
17 class_chart CARD
18     indexing
19         author: "Jakob Melnyk (jmel@itu.dk)";
20     explanation "A card is the representation of the cards within the game."
21     query
22         "What is your card name?",
23         "What is your card number?",
24         "Have you been initialized yet?",
25         "Are you equal to this object?",
26         "Are you and this other card the same?"
27     command
28         "Initialize yourself like this!"
29 end
30
31 cluster_chart CARD_TYPES
32     indexing
33         author: "Jakob Melnyk (jmel@itu.dk)";
34     explanation "The different types of cards that exist."
35     class TREASURE
36         description "A card used to buy new cards."
37     class VICTORY
38         description "A card that grants the points used to win the game."
39     class ACTION
40         description "A card used to help the player buy more cards, get rid of unwanted card
41     class ACTION_ATTACK
42         description "A card that is used to 'attack' other players."
43     class ACTION_REACTION
44         description "A card used to react to opponent attacks."
45     class KINGDOM_VICTORY
46         description "A special kind of victory card."
47 end
48
49 class_chart TREASURE
50     indexing

```

```

51         author: "Jakob Melnyk (jmel@itu.dk)";
52         explanation "A card used to buy new cards."
53         inherit CARD
54     end
55
56     class_chart VICTORY
57         indexing
58             author: "Jakob Melnyk (jmel@itu.dk)";
59             explanation "A card that grants the points used to win the game."
60             inherit CARD
61     end
62
63     class_chart ACTION
64         indexing
65             author: "Jakob Melnyk (jmel@itu.dk)";
66             explanation "A card used to help the player buy more cards, get rid of unwanted cards, etc."
67             inherit CARD
68     end
69
70     class_chart ACTION_ATTACK
71         indexing
72             author: "Jakob Melnyk (jmel@itu.dk)";
73             explanation "A card that is used to 'attack' other players."
74             inherit ACTION
75     end
76
77     class_chart ACTION_REACTION
78         indexing
79             author: "Jakob Melnyk (jmel@itu.dk)";
80             explanation "A card used to react to opponent attacks."
81             inherit ACTION
82     end
83
84     class_chart KINGDOM_VICTORY
85         indexing
86             author: "Jakob Melnyk (jmel@itu.dk)";
87             explanation "A special kind of victory card."
88             inherit VICTORY
89     end
90
91     class_chart CARD_NAME
92         indexing
93             author: "Jakob Melnyk (jmel@itu.dk)";
94             explanation "Represents the values used to refer to the card names."
95         query
96             "May I have the value 'v'?"
97         constraint
98             "The values allowed for this class are exactly one of \
99             \'COPPER', 'SILVER', 'GOLD', 'CURSE', 'ESTATE', 'DUCHY', 'PROVINCE', \
100            \'CELLAR', 'CHAPEL', 'MOAT', 'CHANCELLOR', 'VILLAGE', 'WOODCUTTER', \
101            \'WORKSHOP', 'BUREAUCRAT', 'FEAST', 'GARDENS', 'MILITIA', 'MONEYLENDER', \
102            \'REMODEL', 'SMITHY', 'SPY', 'THIEF', 'THRONE_ROOM', 'COUNCIL_ROOM', 'FESTIVAL', \
103            \'LABORATORY', 'LIBRARY', 'MARKET', 'MINE', 'WITCH', 'ADVENTURER', 'EMPTY', 'BACKSID
104     end
105

```

```

106 class_chart CARD_FACTORY
107     indexing
108         author: "Jakob Melnyk (jmel@itu.dk)";
109     explanation "Factory for producing cards with the correct values."
110     query
111         "Has the factory been set up?",
112         "What cards have been made already?",
113     command
114         "Set up the factory with these cards!",
115         "Give me a card with this name!",
116     constraint
117         "A card that has already been made cannot be made again.",
118 end

1 cluster_chart CARDS
2     indexing
3         author: "Jakob Melnyk (jmel@itu.dk)";
4     explanation "Contains all the cards from a standard Dominion game."
5     class COPPER
6         description "The Copper card."
7     class SILVER
8         description "The Silver card."
9     class GOLD
10        description "The Gold card."
11    class CURSE
12        description "The Curse card."
13    class ESTATE
14        description "The Estate card."
15    class DUCHY
16        description "The Duchy card."
17    class PROVINCE
18        description "The Province card."
19    class CELLAR
20        description "The Cellar card."
21    class CHAPEL
22        description "The Chapel card."
23    class MOAT
24        description "The Moat card."
25    class CHANCELLOR
26        description "The Chancellor card."
27    class VILLAGE
28        description "The Village card."
29    class WOODCUTTER
30        description "The Woodcutter card."
31    class WORKSHOP
32        description "The Workshop card."
33    class BUREAUCRAT
34        description "The Bureaucrat card."
35    class FEAST
36        description "The Feast card."
37    class GARDENS
38        description "The Gardens card."
39    class MILITIA
40        description "The Militia card."
41    class MONEYLENDER

```



```

42         description "The Moneylender card."
43     class REMODEL
44         description "The Remodel card."
45     class SMITHY
46         description "The Smithy card."
47     class SPY
48         description "The Spy card."
49     class THIEF
50         description "The Thief card."
51     class THRONE_ROOM
52         description "The Throne Room card."
53     class COUNCIL_ROOM
54         description "The Council Room card."
55     class FESTIVAL
56         description "The Festival card."
57     class LABORATORY
58         description "The Laboratory card."
59     class LIBRARY
60         description "The Library card."
61     class MARKET
62         description "The Market card."
63     class MINE
64         description "The Mine card."
65     class WITCH
66         description "The Witch card."
67     class ADVENTURER
68         description "The Adventurer card."
69 end
70
71 class_chart COPPER
72     indexing
73         author: "Jakob Melnyk (jmel@itu.dk)";
74     explanation "Worth one coin. Costs no coins."
75     inherit TREASURE
76 end
77
78 class_chart SILVER
79     indexing
80         author: "Jakob Melnyk (jmel@itu.dk)";
81     explanation "Worth two coins. Costs three coins."
82     inherit TREASURE
83 end
84
85 class_chart GOLD
86     indexing
87         author: "Jakob Melnyk (jmel@itu.dk)";
88     explanation "Worth three coins. Costs six coins."
89     inherit TREASURE
90 end
91
92 class_chart CURSE
93     indexing
94         author: "Jakob Melnyk (jmel@itu.dk)";
95     explanation "Worth minus one victory point. Costs no coins."
96 end

```

```

97
98  class_chart ESTATE
99      indexing
100          author: "Jakob Melnyk (jmel@itu.dk)";
101          explanation "Worth one victory point. Costs two coins."
102          inherit VICTORY
103  end
104
105  class_chart DUCHY
106      indexing
107          author: "Jakob Melnyk (jmel@itu.dk)";
108          explanation "Worth three victory points. Costs five coins."
109          inherit VICTORY
110  end
111
112  class_chart PROVINCE
113      indexing
114          author: "Jakob Melnyk (jmel@itu.dk)";
115          explanation "Worth six victory points. Costs eight coins."
116          inherit VICTORY
117  end
118
119  class_chart GARDENS
120      indexing
121          author: "Jakob Melnyk (jmel@itu.dk)";
122          explanation "Worth one victory point for every ten cards in your deck (rounded down) at the
123          inherit KINGDOM_VICTORY
124  end
125
126  class_chart CELLAR
127      indexing
128          author: "Jakob Melnyk (jmel@itu.dk)";
129          explanation "Grants one action. \
130          \Discard any number of cards - draw one card for each card discarded. \
131          \Costs two coins."
132          inherit ACTION
133  end
134
135  class_chart CHAPEL
136      indexing
137          author: "Jakob Melnyk (jmel@itu.dk)";
138          explanation "Trash up to four cards from your hand. Costs two coins."
139          inherit ACTION
140  end
141
142  class_chart CHANCELLOR
143      indexing
144          author: "Jakob Melnyk (jmel@itu.dk)";
145          explanation "Grants two coins. The player may immediately put your deck into your discard pi
146          \Costs three coins."
147          inherit ACTION
148  end
149
150  class_chart VILLAGE
151      indexing

```

```

152         author: "Jakob Melnyk (jmel@itu.dk)";
153         explanation "Grants one card. Grants two actions. Costs three coins."
154         inherit ACTION
155     end
156
157     class_chart WOODCUTTER
158         indexing
159             author: "Jakob Melnyk (jmel@itu.dk)";
160             explanation "Grants one buy. Grants two coins. Costs three coins."
161             inherit ACTION
162         end
163
164     class_chart WORKSHOP
165         indexing
166             author: "Jakob Melnyk (jmel@itu.dk)";
167             explanation "Player gains a card costing up to four coins. Costs three coins."
168             inherit ACTION
169         end
170
171     class_chart FEAST
172         indexing
173             author: "Jakob Melnyk (jmel@itu.dk)";
174             explanation "Player trashes this card. Gain a card costing up to five coins. Costs 4 coins."
175             inherit ACTION
176         end
177
178     class_chart MONEYLENDER
179         indexing
180             author: "Jakob Melnyk (jmel@itu.dk)";
181             explanation "Player trashes a Copper card from his/her hand. \
182                 \If the player does so, he is granted three coins. Costs four coins."
183             inherit ACTION
184         end
185
186     class_chart REMODEL
187         indexing
188             author: "Jakob Melnyk (jmel@itu.dk)";
189             explanation "Player trashes a card from his/her hand. Player gains a card costing up to two
190                 \than the trashed card. Costs 4 coins."
191             inherit ACTION
192         end
193
194     class_chart SMITHY
195         indexing
196             author: "Jakob Melnyk (jmel@itu.dk)";
197             explanation "Grants three cards. Costs four coins."
198             inherit ACTION
199         end
200
201     class_chart THRONE_ROOM
202         indexing
203             author: "Jakob Melnyk (jmel@itu.dk)";
204             explanation "Player chooses an Action card in his/her hand. That card is played twice. Costs
205             inherit ACTION
206         end

```

207

208 class\_chart COUNCIL\_ROOM

209 indexing

210 author: "Jakob Melnyk (jmel@itu.dk)";

211 explanation "Grants four cards. Grants one buy. All other players are granted one card. Cost

212 inherit ACTION

213 end

214

215 class\_chart FESTIVAL

216 indexing

217 author: "Jakob Melnyk (jmel@itu.dk)";

218 explanation "Grants two actions. Grants one buy. Grants two coins. Costs five coins."

219 inherit ACTION

220 end

221

222 class\_chart LABORATORY

223 indexing

224 author: "Jakob Melnyk (jmel@itu.dk)";

225 explanation "Grants two cards. Grants one action. Costs five coins."

226 inherit ACTION

227 end

228

229 class\_chart LIBRARY

230 indexing

231 author: "Jakob Melnyk (jmel@itu.dk)";

232 explanation "Player draws until he/she has seven cards in hand. Player may set aside any act

233 \ drawn this way; discard the set aside cards after the Player is fi

234 inherit ACTION

235 end

236

237 class\_chart MARKET

238 indexing

239 author: "Jakob Melnyk (jmel@itu.dk)";

240 explanation "Grants one card. Grants one action. Grants one buy. Grants one coin. Costs five

241 inherit ACTION

242 end

243

244 class\_chart MINE

245 indexing

246 author: "Jakob Melnyk (jmel@itu.dk)";

247 explanation "Player trashes a Treasure card from his/her hand. Player gains a treasure card

248 \up to three coins more. Costs five coins."

249 inherit ACTION

250 end

251

252 class\_chart ADVENTURER

253 indexing

254 author: "Jakob Melnyk (jmel@itu.dk)";

255 explanation "Player reveals cards from his/her deck until two Treasure have been revealed. \

256 \Player puts the two Treasure cards into hand and discard the other

257 inherit ACTION

258 end

259

260 class\_chart BUREAUCRAT

261 indexing

```

262         author: "Jakob Melnyk (jmel@itu.dk)";
263         explanation "Player gains a silver card on top of deck. Each other Player reveals a Victory
264             \and puts it on top of his deck (or reveals a hand with no Victory
265         inherit ACTION_ATTACK
266     end
267
268     class_chart MILITIA
269         indexing
270             author: "Jakob Melnyk (jmel@itu.dk)";
271             explanation "Grants two coins. Each other player discards down to three cards in his/her han
272             inherit ACTION_ATTACK
273     end
274
275     class_chart SPY
276         indexing
277             author: "Jakob Melnyk (jmel@itu.dk)";
278             explanation "Grants one card. Grants one action. Each Player (including the active Player) r
279             \and the active Player decides to either put the card back or discar
280             inherit ACTION_ATTACK
281     end
282
283     class_chart THIEF
284         indexing
285             author: "Jakob Melnyk (jmel@itu.dk)";
286             explanation "Each other Player reveals the top two cards of his/her deck. If any Treasure ca
287             \ the active Player can choose to trash one of them. The active play
288             \ The other revealed cards are discarded. Costs four coins."
289             inherit ACTION_ATTACK
290     end
291
292     class_chart WITCH
293         indexing
294             author: "Jakob Melnyk (jmel@itu.dk)";
295             explanation "Grants two cards. Each other player gains a Curse card. Costs five coins."
296             inherit ACTION_ATTACK
297     end
298
299     class_chart MOAT
300         indexing
301             author: "Jakob Melnyk (jmel@itu.dk)";
302             explanation "Grants two cards. When another Player plays an Attack card and this card is in
303             \ this card makes you unaffected by that Attack. Costs two coins."
304             inherit ACTION_REACTION
305     end

```

  

```

1  cluster_chart GUI
2  indexing
3      author: "Frederik Lysgaard (frly@itu.dk)";
4  explanation " The graphical representation part of the project"
5  class GAMECLASS description " The game class"
6  class DECKZONE description " The deck class"
7  class ACTIONZONE description " The card class"
8  class DISCARDZONE description " The discardzone class"
9  class HANDZONE description " The handzone class"
10 class SUPPLYZONE description " The supplyzone class"

```

```

11  class BUTTONSPRITE description " The buttonsprite class"
12  class CARDSprite description " The cardsprite class"
13  class PROGRAM description " The program class"
14  class GUICONSTANTS description " The constant class"
15  class GUIINTERFACE description " The interface class for the gui"
16  end
17
18  class_chart DECKZONE
19    indexing
20      author: "Frederik Lysgaard (frly@itu.dk)";
21      explanation " responsible for representing the deck"
22      command
23        "Draw the content!"
24    end
25
26  class_chart BUTTONSPRITE
27    indexing
28      author: "Frederik Lysgaard (frly@itu.dk)";
29      explanation " the basic class which all graphical objects should inherit from"
30      command
31        "Draw the content!"
32    end
33
34  class_chart CARDSprite
35    indexing
36      author: "Frederik Lysgaard (frly@itu.dk)";
37      explanation " responsible for representing the cards"
38      query
39        "Is this cardsprite equal to this cardsprite?"
40      command
41        "Draw the content!"
42    end
43
44  class_chart PROGRAM
45    indexing
46      author: "Frederik Lysgaard (frly@itu.dk)";
47      explanation " responsible for executing the game"
48      command
49        " Run a clinet!",
50        " Run a Host!",
51        " Start the GUI!",
52    end
53
54  class_chart GAMECLASS
55    indexing
56      author: "Frederik Lysgaard (frly@itu.dk)";
57      explanation " responsible for creating the initial GUI with the components from the other classes"
58      command
59        " Initialize the content!",
60        " Load the content!",
61        " Unload the content!",
62        " Update the game!",
63        " Draw the content!",
64    end
65

```

```

66 class_chart GUICONSTANTS
67   indexing
68     author: "Frederik Lysgaard (frly@itu.dk)";
69     explanation " responsible for keeping all the constants used in GUI i one place"
70 end
71
72 class_chart GUIINTERFACE
73   indexing
74     author: "Frederik Lysgaard (frly@itu.dk)";
75     explanation " responsible for the interface between the GUI and the Controller"
76     command
77       "Run the game!",
78       "Draw the hand!",
79       "Draw the actionzone!",
80       "Draw the discardzone!",
81       "Draw the deck!",
82       "Set actions!",
83       "Set buys!",
84       "Set coins!",
85       "Set endgame!",
86       "Set the turn!",
87       "Set the phase!",
88       "Set the playernumber!",
89       "Make the supplyzone!",
90 end

1  cluster_chart NETWORK_CLUSTER
2  indexing
3    author: "Christian 'Troy' Jensen, chrj@itu.dk";
4  explanation "The part of the program responsible for running the network"
5  class CLIENT description "A network client"
6  class SERVER description "A network server"
7  class CONNECTION description "A connection between a server and a client"
8  class NETWORKING_INTERFACE description "A network interface"
9
10 end
11
12 class_chart CLIENT
13 indexing
14   author: "Christian 'Troy' Jensen, chrj@itu.dk";
15   explanation "Represents a player in a game of Dominion, one for each player"
16   query
17     "Can I have the connection for this client?",
18   command
19     "Begin recieving more messages!",
20 end
21
22 class_chart SERVER
23 indexing
24   author: "Christian 'Troy' Jensen, chrj@itu.dk";
25   explanation "Responsible for managing the clients of a game, only one per game"
26   query
27     "Can I have the IP of the server?",
28     "Can I have a list of the known clients",
29   command

```

```

30         "Start the server!",
31         "Send this as a system message to this client!",
32         "Send this as a system message to all clients!",
33         "Forward this message!",
34     end
35
36     class_chart CONNECTION
37     indexing
38         author: "Christian 'Troy' Jensen, chrj@itu.dk";
39     explanation "Responsible for holding all the information on a client that a server has, one for each"
40     query
41         "Can I have the IP of the client",
42         "Can I have the Id of the Client",
43     command
44         "Send this message!",
45         "Begin recieving more messages!",
46     end
47
48     class_chart NETWORKING_INTERFACE
49     indexing
50         author: "Christian 'Troy' Jensen, chrj@itu.dk";
51     explanation "The outward face of a networking session, keeps track of a client and maybe a server"
52     query
53         "Is this interface running a server?",
54         "Can I have the IP of the server?",
55     command
56         "This is the number of clients!",
57         "Send this message, and you better give me some answers!",
58         "Send this message",
59     end
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



```

25         end
26         GetScores : SEQUENCE[INTEGER]
27
28         --Cmmands
29         SetActivePlayer
30             -> player : PLAYER
31             require playerNumber >= 1 and playerNumber <= NumberOfPlayers and player /=
32         end
33         StartActionPhase
34             require InActionPhase = false and InBuyPhase = false
35             ensure InActionPhase = true and InBuyPhase = false
36         end
37         EndActionPhase
38             require InActionPhase = true and InBuyPhase = false
39             ensure InActionPhase = false and InBuyPhase = false and NumberOfActionsLeft
40         end
41         StartBuyPhase
42             require InActionPhase = false and InBuyPhase = false
43             ensure InActionPhase = false and InBuyPhase = true
44         end
45         EndBuyPhase
46             require InActionPhase = false and InBuyPhase = true
47             ensure InActionPhase = false and InBuyPhase = false and NumberOfBuysLeft = 0
48         end
49         DoCleanUp
50             require InActionPhase = false and InBuyPhase = false
51         end
52         IncreaseActions
53             -> amount : INTEGER
54             require amount + NumberOfActionsLeft >= 0
55         end
56         IncreaseBuys
57             -> amount : INTEGER
58             require amount + NumberOfBuysLeft >= 0
59         end
60         IncreaseCoins
61             -> amount : INTEGER
62             require amount + NumberOfCoinsLeft >= 0
63         end
64         PlayerGainsCard
65             -> player : PLAYER
66             -> card : CARD_NAME
67             require player member_of GetPlayers and player /= void
68         end
69     end
70     class PLAYER
71         indexing
72             author: "Jakob Melnyk (jmel@itu.dk)";
73         feature
74             --Queries
75             GetAllCards : SET[CARD]
76             GetDeckSize : NATURAL
77             GetDiscardSize : NATURAL
78             GetHand : SEQUENCE[CARD]
79             GetPlayerNumber : NATURAL

```

```

80      GetTopOfDiscard : CARD
81          require GetDiscardSize /= 0
82      end
83      GetTopOfDeck : CARD
84          require GetDiscardSize /= 0
85      end
86      GetPlayed : SEQUENCE[CARD]
87      GetTemporaryZone : SEQUENCE[CARD]
88
89      --Commands
90      MoveFromZoneToTemporary
91          -> zone : ZONE
92          require (zone = DECK or zone = DISCARD) and (zone = DECK -> (GetDeckSize = 0
93              and (zone = DISCARD -> (GetDiscardSize /= 0))
94
95              ensure GetTemporaryZone.Count = old GetTemporaryZone.Count + 1          and (z
96                  and (zone = DISCARD -> GetDiscardSize = old GetDiscardSize -
97      end
98      MoveFromHandToTemporary
99          -> card : CARD
100          require GetHand.Contains(card) = false and card /= void
101          ensure GetHand.Contains(card) = false and GetTemporaryZone.Contains(card)
102      end
103      MoveFromTemporary
104          -> card : CARD
105          -> zone : ZONE
106          require (zone = DECK or zone = DISCARD or zone = HAND or zone = PLAYED) and
107          ensure (GetTemporaryZone.Count = old GetTemporaryZone.Count - 1) and
108              (zone = DECK -> GetDeckSize = old GetDeckSize +1 ) and
109              (zone = DECK -> GetTopOfDeck = old GetTemporaryZone.Get(old
110              (zone = DISCARD -> GetDiscardSize = old GetDiscardSize + 1)
111              (zone = DISCARD -> GetTopOfDiscard = old GetTemporaryZone.Ge
112              (zone = HAND -> GetHand.Count = old GetHand.Count + 1) and G
113              (zone = HAND -> GetHand.Get(GetHand.Count - 1) = old GetTemp
114              (zone = PLAYED -> GetPlayed.Count = old Played.Count + 1) an
115              (zone = PLAYED -> GetPlayed.Get(Played.Count - 1) = old GetT
116      end
117      AddCardToZone
118          -> card : CARD
119          -> zone : ZONE
120          require (zone = DECK or zone = DISCARD or zone = HAND or zone = PLAYED) and
121          ensure GetAllCards.Contains(card) and
122              (zone = HAND -> GetHand.Get(GetHand.Count - 1) = card) and
123              (zone = HAND -> GetHand.Count = old GetHand.Count + 1) and
124              (zone = PLAYED -> GetPlayed.Get(GetPlayed.Count - 1) = card)
125              (zone = PLAYED -> GetPlayed.Count = old GetPlayed.Count + 1)
126              (zone = DISCARD -> GetDiscardSize = old GetDiscardSize + 1)
127              (zone = DISCARD -> GetTopOfDiscard = card) and
128              (zone = DECK -> GetDeckSize = old GetDeckSize +1 ) and
129              (zone = Deck -> GetTopOfDeck = card)
130      end
131      RemoveCardFromZone
132          -> card : CARD
133          -> zone : ZONE
134          require (zone = DECK or zone = DISCARD or zone = HAND or zone = PLAYED) and

```

```

135         and (zone = HAND -> GetHand.Contains(card))
136         and (zone = PLAYED -> GetPlayed.Contains(card))
137         and (zone = DECK -> (GetDeckSize = 0 and GetDiscardSize = 0))
138         and (zone = DISCARD -> GetDiscardSize /= 0)
139     ensure GetAllCards.Contains(card) = false and
140         (zone = HAND -> GetHand.Contains(card) = false) and
141         (zone = HAND -> GetHand.Count = old GetHand.Count - 1) and
142         (zone = PLAYED -> GetPlayed.Contains(card) = false) and
143         (zone = PLAYED -> GetPlayed.Count = old GetPlayed.Count - 1)
144         (zone = DISCARD -> GetDiscardSize = old GetDiscardSize - 1)
145         (zone = DECK -> GetDeckSize = old GetDeckSize - 1)
146     end
147     DrawCards
148         -> amount : NATURAL
149     DrawCard
150         require GetDeckSize + GetDiscardSize /= 0
151         ensure GetHand.Count = old GetHand.Count + 1
152     end
153
154     --Invariant: A card cannot be in the DECK, DISCARD, HAND or PLAYED zones of a player
155     --         if it is not in the 'ALL CARDS'.
156 end
157 class ZONE
158     indexing
159         author: "Jakob Melnyk (jmel@itu.dk)";
160     feature
161         --Queries
162         value : STRING
163             ensure Result = "DECK" or Result = "DISCARD" or Result = "HAND" or Result =
164         end
165         --Commands
166     end
167 end

1 static_diagram CARD_TYPES_CLUSTER
2     component
3         class TREASURE
4             indexing
5                 author: "Jakob Melnyk (jmel@itu.dk)";
6                 inherit CARD
7         end
8
9         class VICTORY
10             indexing
11                 author: "Jakob Melnyk (jmel@itu.dk)";
12                 inherit CARD
13         end
14
15         class ACTION
16             indexing
17                 author: "Jakob Melnyk (jmel@itu.dk)";
18                 inherit CARD
19         end
20
21         class ACTION_ATTACK

```

```

22         indexing
23             author: "Jakob Melnyk (jmel@itu.dk)";
24         inherit ACTION
25     end
26
27     class ACTION_REACTION
28         indexing
29             author: "Jakob Melnyk (jmel@itu.dk)";
30         inherit ACTION
31     end
32
33     class KINGDOM_VICTORY
34         indexing
35             author: "Jakob Melnyk (jmel@itu.dk)";
36         inherit VICTORY
37     end
38 end

```

```

1  static_diagram CARD_CLUSTER
2  component
3      class CARD
4          indexing
5              author: "Jakob Melnyk (jmel@itu.dk)";
6          feature
7              --Queries
8              EqualsOtherObj : BOOLEAN
9                  -> obj : VALUE -- Object in C#.
10             EqualsOtherCard : BOOLEAN
11                 -> other : CARD
12             GetName : CARD_NAME
13             GetNumber : NATURAL
14             SetUp : BOOLEAN
15             --Commands
16             Initialize
17                 -> name : CARD
18                 -> number : NATURAL
19                 require SetUp = false
20                 ensure  SetUp = true
21         end
22     end
23
24     class CARD_FACTORY
25         indexing
26             author: "Jakob Melnyk (jmel@itu.dk)";
27         feature
28             --Queries
29             SetUp : BOOLEAN
30             CreatedCards : SET[CARD]
31             CardsMade : TABLE[CARD_NAME, NATURAL] --private
32
33             --Commands
34             CreateCard : CARD
35                 -> Card : CARD_NAME
36                 ensure Result.GetName = CARD_NAME
37         end

```

```

38         SetUpCards
39             -> cards : COLLECTION[CARD_NAME]
40             require SetUp = false and cards /= void
41             ensure SetUp = true
42         end
43         --Invariant commented because I could not get it to compile, but below is a rough id
44         --for_all c member_of CreatedCards it_holds c.GetNumber < CardsMade.get(c.GetName)
45     end
46
47     class CARD_NAME
48         indexing
49             author: "Jakob Melnyk (jmel@itu.dk)";
50         feature
51             --Queries
52             value : STRING --This looks very awkward, but we felt it best described what we wanted
53             ensure Result = "COPPER" or Result = "GOLD" or Result = "SILVER" or
54             Result = "CURSE" or Result = "ESTATE" or Result = "DUCHY" or Result = "PROVINCE" or
55             Result = "CELLAR" or Result = "CHAPEL" or Result = "MOAT" or Result = "CHANCE" or
56             Result = "VILLAGE" or Result = "WOODCUTTER" or Result = "WORKSHOP" or
57             Result = "BUREAUCRAT" or Result = "FEAST" or Result = "GARDENS" or Result =
58             Result = "MONEYLENDER" or Result = "REMODEL" or Result = "SMITHY" or Result =
59             Result = "THIEF" or Result = "THRONE_ROOM" or Result = "COUNCIL_ROOM" or Result =
60             Result = "LABORATORY" or Result = "LIBRARY" or Result = "MARKET" or Result =
61             Result = "EMPTY" or Result = "BACKSIDE"
62         end
63         --Commands
64     end
65 end

1  static_diagram CARDS_CLUSTER
2      component
3          class COPPER
4              indexing
5                  author: "Jakob Melnyk (jmel@itu.dk)";
6              inherit TREASURE
7          end
8
9          class SILVER
10             indexing
11                 author: "Jakob Melnyk (jmel@itu.dk)";
12             inherit TREASURE
13         end
14
15         class GOLD
16             indexing
17                 author: "Jakob Melnyk (jmel@itu.dk)";
18             inherit TREASURE
19         end
20
21         class CURSE
22             indexing
23                 author: "Jakob Melnyk (jmel@itu.dk)";
24         end
25
26         class ESTATE

```

```

27         indexing
28         author: "Jakob Melnyk (jmel@itu.dk)";
29         inherit VICTORY
30     end
31
32     class DUCHY
33         indexing
34         author: "Jakob Melnyk (jmel@itu.dk)";
35         inherit VICTORY
36     end
37
38     class PROVINCE
39         indexing
40         author: "Jakob Melnyk (jmel@itu.dk)";
41         inherit VICTORY
42     end
43
44     class GARDENS
45         indexing
46         author: "Jakob Melnyk (jmel@itu.dk)";
47         inherit KINGDOM_VICTORY
48     end
49
50     class CELLAR
51         indexing
52         author: "Jakob Melnyk (jmel@itu.dk)";
53         inherit ACTION
54     end
55
56     class CHAPEL
57         indexing
58         author: "Jakob Melnyk (jmel@itu.dk)";
59         inherit ACTION
60     end
61
62     class CHANCELLOR
63         indexing
64         author: "Jakob Melnyk (jmel@itu.dk)";
65         inherit ACTION
66     end
67
68     class VILLAGE
69         indexing
70         author: "Jakob Melnyk (jmel@itu.dk)";
71         inherit ACTION
72     end
73
74     class WOODCUTTER
75         indexing
76         author: "Jakob Melnyk (jmel@itu.dk)";
77         inherit ACTION
78     end
79
80     class WORKSHOP
81         indexing

```

```

82             author: "Jakob Melnyk (jmel@itu.dk)";
83         inherit ACTION
84     end
85
86     class FEAST
87         indexing
88             author: "Jakob Melnyk (jmel@itu.dk)";
89         inherit ACTION
90     end
91
92     class MONEYLENDER
93         indexing
94             author: "Jakob Melnyk (jmel@itu.dk)";
95         inherit ACTION
96     end
97
98     class REMODEL
99         indexing
100             author: "Jakob Melnyk (jmel@itu.dk)";
101         inherit ACTION
102     end
103
104     class SMITHY
105         indexing
106             author: "Jakob Melnyk (jmel@itu.dk)";
107         inherit ACTION
108     end
109
110     class THRONE_ROOM
111         indexing
112             author: "Jakob Melnyk (jmel@itu.dk)";
113         inherit ACTION
114     end
115
116     class COUNCIL_ROOM
117         indexing
118             author: "Jakob Melnyk (jmel@itu.dk)";
119         inherit ACTION
120     end
121
122     class FESTIVAL
123         indexing
124             author: "Jakob Melnyk (jmel@itu.dk)";
125         inherit ACTION
126     end
127
128     class LABORATORY
129         indexing
130             author: "Jakob Melnyk (jmel@itu.dk)";
131         inherit ACTION
132     end
133
134     class LIBRARY
135         indexing
136             author: "Jakob Melnyk (jmel@itu.dk)";

```

```

137             inherit ACTION
138         end
139
140     class MARKET
141         indexing
142             author: "Jakob Melnyk (jmel@itu.dk)";
143         inherit ACTION
144     end
145
146     class MINE
147         indexing
148             author: "Jakob Melnyk (jmel@itu.dk)";
149         inherit ACTION
150     end
151
152     class ADVENTURER
153         indexing
154             author: "Jakob Melnyk (jmel@itu.dk)";
155         inherit ACTION
156     end
157
158     class BUREAUCRAT
159         indexing
160             author: "Jakob Melnyk (jmel@itu.dk)";
161         inherit ACTION_ATTACK
162     end
163
164     class MILITIA
165         indexing
166             author: "Jakob Melnyk (jmel@itu.dk)";
167         inherit ACTION_ATTACK
168     end
169
170     class SPY
171         indexing
172             author: "Jakob Melnyk (jmel@itu.dk)";
173         inherit ACTION_ATTACK
174     end
175
176     class THIEF
177         indexing
178             author: "Jakob Melnyk (jmel@itu.dk)";
179         inherit ACTION_ATTACK
180     end
181
182     class WITCH
183         indexing
184             author: "Jakob Melnyk (jmel@itu.dk)";
185         inherit ACTION_ATTACK
186     end
187
188     class MOAT
189         indexing
190             author: "Jakob Melnyk (jmel@itu.dk)";
191         inherit ACTION_REACTION

```



```

192         end
193     end

1  static_diagram GUI
2  component
3      class GuiInterface
4          indexing
5              author: "Christian 'Troy' Jensen, chrj@itu.dk";
6          feature
7              --Commands
8              Run
9              DrawHand
10                 -> cards : SEQUENCE[CARD]
11             DrawAction
12                 -> cards : SEQUENCE[CARD]
13             DrawDiscard
14                 -> card : CARD
15             DrawDeck
16                 -> filled : bool --Whether there are any cards in the deck
17             SetAction
18                 -> number : INTEGER
19             SetBuys
20                 -> number : INTEGER
21             SetCoins
22                 -> number : INTEGER
23             EndGame
24                 -> playerId : INTEGER
25             YourTurn
26                 -> yourTurn : BOOLEAN
27             SetPhase
28                 -> phase : INTEGER
29             UsedCards
30                 -> cards : SEQUENCE[CARD]
31             SetPlayerNumber
32                 -> id : INTEGER
33         end
34     end

1  --NOTICE: This network design is based heavily on code I got
2  --from Simon Henriksen (shen@itu.dk) and where there are similarities
3  --between our code, he deserves the full credit for its design.
4
5  --Receiving
6
7  static_diagram NETWORK_CLUSTER
8  component
9      class CONNECTION
10         indexing
11             author: "Christian 'Troy' Jensen, chrj@itu.dk";
12         feature
13             --Queries
14             GetClientIp : IPADDRESS --C# object
15             GetId : INTEGER
16             --Commands

```

```

17             Send
18                 -> message : STRING
19             BeginRecieve
20
21         end
22
23     class SERVER
24         indexing
25             author: "Christian 'Troy' Jensen, chrj@itu.dk";
26         feature
27             --Queries
28             GetIp : IPADDRESS
29             GetClientList : SEQUENCE[CONNECTION]
30
31             --Commands
32             Start
33             SystemMessageToClient --Sent to a particular client
34                 -> message : STRING
35                 -> CONNECTION : CONNECTION --C# object
36             SystemMessageToAll --Sent to all clients
37                 -> message : STRING
38             ForwardMessage
39                 -> message : STRING
40                 -> clientId : INTEGER
41                 -> type : MESSAGE TYPE
42
43         end
44
45     class CLIENT
46         indexing
47             author: "Christian 'Troy' Jensen, chrj@itu.dk";
48         feature
49             --Queries
50             GetComm : SOCKET --C# object
51
52             --Commands
53             BeginReceive
54
55         end
56
57     class NETWORKCONST
58         indexing
59             author: "Christian 'Troy' Jensen, chrj@itu.dk";
60         feature
61             --All these are constants
62             GetEncoder : UTF8ENCODING --C# object
63             GetPort : INTEGER
64             GetBufferSize : INTEGER
65         end
66
67     class NETWORKINGINTERFACE
68         indexing
69             author: "Christian 'Troy' Jensen, chrj@itu.dk";
70         feature
71             --Queries

```

```

72         IsServer : BOOLEAN
73         GetServerIP : STRING
74         SetNumberOfClients
75             -> TotalClients : INTEGER
76     --Commands
77         SendTurnMessage : SEQUENCE[STRING] --Responses from the other player
78             -> Message : STRING
79         SendPreGameMessage
80             -> Message : STRING
81     end
82 end

```

## References

- [1] <http://www.riograndegames.com/games.html?id=278>
- [2] [http://www.riograndegames.com/uploads/Game/Game\\_278\\_gameRules.pdf](http://www.riograndegames.com/uploads/Game/Game_278_gameRules.pdf)
- [3] Simon Henriksen shen@itu.dk