Department of Computer Science

Magnus Bohman
Fredric Hellberg

Development of a Prototype to Determine an Individual's Utility Function

Master's Thesis

2005:07

Development of a Prototype to Determine an Individual's Utility Function

Magnus Bohman
Fredric Hellberg

This thesis is submitted in partial fulfillment of the requirements for the Masters degree in Computer Science. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Magnus Bohman

Fredric Hellberg

Approved, 2005-06-07

Opponent: Thijs Holleboom

Advisor: Albin Zuccato

Examiner: Donald F. Ross

Abstract

Theories from the fields of economics, business management and decision analysis allow for the assessment of risk attitudes. Using games of chance it is possible to get an insight in the risk attitude of a decision maker. In this thesis an application is presented, which assesses risk attitudes by letting a decision maker play such games of chance.

The games indicate how the decision maker behaves in risky situations versus certain situations. The game results are then transformed into a utility function, using regression analysis. The utility function can be used by other applications to prioritise risk situations based on the assessed risk attitude of the decision maker.

The thesis also discuss the development of the application and the experience derived from the application of extreme programming as development approach.

Acknowledgements

We want to thank our supervisor Albin Zuccato, for steering us in the right way and helping us throughout this project. We also want to thank Magnus's mother, Ylva, for her help checking the grammar and spelling of the thesis.

I, Fredric want to thank my children, Leon and Alva for great understanding when I was tired in the mornings because of late nights writing this thesis. I also want to thank my father, my brother and all my friends for their great support and understanding during the thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"What you risk reveals what you value."
– Jeanette Winterson (1961-)

Every day people make decisions. Some decisions are small, such as the decision whether to buy a soda or an ice-cream. These decisions are often quite easy to make. Other decisions are big, such as decisions made in politics or in business management. These decisions are often hard to make because they often have great impact.

There are always risks involved in the decision making process, but the significance of the risks varies depending on the decision maker's attitude towards them. The decision maker's attitude towards risks differ depending on the situation.

If a decision maker's risk attitude somehow could be determined, people at a lower level in the decision making hierarchy could benefit from an insight in how the decision maker would value different aspects of a decision, in regards to risk attitude.

The theories coming from the field of economics and business management, allow for the assessment of risk attitude. By letting a decision maker play games, it is possible to determine the risk attitude, or utility function of that individual.

In French [7] games are played with the help of an interviewer, asking questions for the decision maker to answer. If the interviewer could be replaced by a computer program, which asks the decision maker the same questions, the risk attitude could be automatically determined.

The application developed in this thesis takes the first step towards an application that ultimately will enable users to assess security risks, using the pre-determined risk attitude, according to the Modified Mean Value approach, put forward by Zuccato [16]. This goal proved to be too time consuming, therefore the application was developed with the goal to determine risk attitudes and utility functions. The application was developed using the Extreme Programming (XP) software development method. As part of the thesis, the development of the application is described.

This thesis was written at the Department of Computer Science at Karlstad University. This project was initiated by Albin Zuccato, who was the owner of the project and also had the role of the XP customer during the application development. The project owner decided that the application was to be named Mmv, after his modified mean value approach.

1.1 Objectives

The main objective is the creation of a Java application, using the extreme programming method. The application should try to determine an individual's utility function.

The secondary objective is to compile a documentation covering both the development phase as well as the theories needed to determine a utility function.

1.2 Target Audience

This thesis is directed to people interested in risk analysis. The reader is assumed to have some mathematical and statistical knowledge. As this thesis also describes the development

of a Java application, it is assumed that the reader has some knowledge of both Java and application development.

1.3 Scientific Method

The scientific methods used in this thesis are both experimental and analytic. The experimental part is the creation of an application to determine risk attitudes described in chapter 2. The analytic part of the thesis is the literature study to search for information to obtain enough knowledge on how to determine risk attitudes.

1.4 Outline

This thesis begins with a description of its background and purpose. In Chapter 2, the decision and risk theories are introduced and a method for assessment of a person's risk attitude is covered.

In Chapter 3 a method for approximating a mathematical model to a set of values is described. This method is called regression analysis and converts the results from the risk attitude assessment method into a continuous function.

Chapter 4 covers the environment, software model and tools used during development of the application mentioned in the main objective for this thesis. In Chapter 5 the development of the two prototypes that were developed prior to the main application is described.

Chapters 6 to 9 describe the development of the main application. Chapter 6 is a common chapter for the three iterations that were produced during the development. Chapter 7 describes the first iteration and chapters 8 and 9 describe the next two iterations. Chapter 10 concludes the thesis and contains a discussion of experiences as well as a presentation of further works.

Chapter 2

Decision and Risk Analysis Theory

"When it is not necessary to make a decision, it is necessary not to make a decision."
– Lord Falkland (1610-1643)

2.1 Introduction

This chapter introduces decision and risk analysis theories. It begins with a definition of what a risk is and then tries, by means of a few examples, to explain the meaning of different risk attitudes. After risks and risk attitudes are explained, the method used in this thesis to assess a person's risk attitude is explained. Then the chapter continues with an explanation of the utility function, which is the representation of a person's attitude toward risks. The utility function is the result given from the method for assessment of risk attitudes. Throughout this chapter some mathematical reasoning is done, which is explained in more detail in chapter 3.

2.2 Risks

According to Frosdick [8], the British Standard 4778 defines risk as "a combination of the probability, or frequency, of occurrence of a defined hazard and the magnitude of the consequences of the occurrence". A consequence can for example be a monetary loss or, in the area of computer security, be a network intrusion or the loss of critical data. Consequences with small magnitudes that occur frequently (high probability of occurrence) are often preferred to consequences with big magnitudes that seldom occur (lower probability of occurrence). Taking a risk, is hereafter thought of as making a decision with uncertain outcome.

People's attitude toward risks differ. A person can be **risk averse**, **risk neutral** or **risk prone**. A person that is risk averse avoids taking risks, whereas a risk prone person is more willing to take risks. A person that is risk neutral is indifferent whether to take a risk or to decide for the certain alternative as long as they are on the same preference curve [7]. Indifference means that a person having two options, a and b, does not prefer a over b or b over a.

Although a person generally is either more or less risk prone or risk averse, a risk averse person can become more risk prone when the gain of taking the risk outweighs the eventual loss. A risk prone person can become more risk averse when the eventual loss outweighs the gain of taking the risk. This can be further explained with an example.

2.2.1 Example

A decision maker (hereafter the abbreviation DM will be used), is faced with a scenario as in figure 2.1. The probability for each outcome is defined by p which can be any value between 0 and 1. In this scenario a DM in a certain situation can decide to increase security for a certain cost or the DM can choose to do nothing.

If not increasing security, there may be a security breach with an estimated cost and a

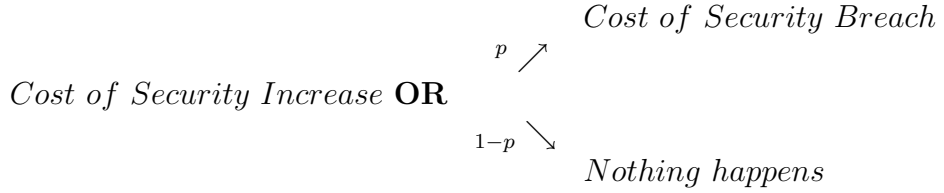

Figure 2.1: Risk scenario

probability of p or there may be a situation where nothing will happen with a probability of $1 - p$.

This scenario is an artificial scenario, in the sense that it assumes that increased security results in no security breaches in a foreseeable future. In the real world this assumption is not true, since there is no guarantee that a security increase would actually stop all breaches.

A solely risk averse DM would increase security instead of taking the possibility that a breach would occur (although the mean value would suggest different). As soon as the occurrence probability (p) of a breach is higher than zero a solely risk averse DM does that decision. If a DM is solely risk prone instead, the decision would be the opposite. This means that if the probability is less than one for a breach, then a solely risk prone DM would do the decision not to increase security. A risk neutral DM faced with the same scenario is indifferent between increasing security or not, if the benefit is the same.

A DM is usually not solely risk averse or solely risk prone, but the risk attitude is somewhere in between. Every DM has a threshold at which the risk attitude changes from risk averse to risk neutral and from risk neutral to risk prone, or vice versa. If a DM is faced with the scenario in figure 2.1, not increasing security would save the DM the cost of increased security, but that also results in a less secure situation. Depending on the probability p, the average cost of a security breach may be lower or higher than the cost for increased security.

Depending on the cost of the security increase, the average cost of a breach and the consequences of a breach, a former risk averse DM can be less risk averse or even become

risk prone.

Similarly, a risk prone DM can be less prone if, in the event of a breach, the consequences are deemed to high. For instance, if the reputation of a network security company is damaged due to an intrusion, then the resulting loss in sales might be higher than the increased security costs.

A risk neutral DM is indifferent between the upgrading cost and the cost of an eventual breach. The DM must base its decision on other preferences, which are out of the scope of this thesis to discuss.

2.3 Games

There are several methods used for determining a person's risk attitude. For instance: certain equivalence (CE) and probability equivalence (PE) [5]. The PE method, which is used in this thesis to measure a person's risk attitude, was a requirement from the project owner[1]. The PE method will be described in this section.

To determine a person's attitude toward risks the person is told to play a number of games. In this context, the word game is defined as letting the decision maker or player answer a question. There are actually no real game-playing going on in the sense that one normally would think of playing games.

In each game the player is offered the choice between getting a sum of money for certain or a lottery ticket. The goal of the game is to find out when the player is indifferent between taking the money for certain and playing the lottery.

Depending on when the player is indifferent between taking the money or playing the lottery, for a number of different scenarios a risk attitude can be approximated. The approximation is then converted into a function called the utility function.

According to Gordon [10], the *utility* is a numerical value associated to each monetary

[1] As mentioned in the introduction the project owner was Albin Zuccato

gain and loss, indicating the decision makers desirability or undesirability of these monetary gains or losses.

The PE-game and CE-game look like equation 2.1, where p and $1-p$ are the probabilities for the monetary outcomes x_1 and x_2, respectively and x_c is a sum that the player certainly gets. The symbol '$\sim$' is used to denote indifference.

$$\underbrace{x_c}_{certain} \sim \underbrace{\begin{cases} (p, x_1) \\ (1-p, x_2) \end{cases}}_{lottery}, \ where\ x_2 < x_c < x_1 \tag{2.1}$$

The difference between the two games is how they are played. In a PE game the player is asked to set the probability p at which the player is indifferent ($\sim$) between playing the lottery and certainly getting x_c. In a CE game the probability p is given and the player sets x_c.

From a security point of view, the two games, PE and CE are different in that PE games tend to generate a more risk averse result than CE games. In this thesis it is assumed that a risk averse result is preferred over a risk prone result. If a risk averse person is playing a CE game, the result may be that the person seems to be more risk prone than it really is. When the result from playing is used as the basis for decision making in scenarios concerning security, the consequences of this may be expensive due to the highly volatile and uncertain character of information systems.[5]

For instance, think of the scenario in figure 2.1, where a person that in reality is risk averse and normally would choose to increase security, would make its decision based on playing games. It is more likely that the person would increase the security based on the result from a PE game than from a CE game. A risk prone person would, when playing a PE game, seem more risk averse and may increase the security. To increase the security, for instance by upgrading a firewall, is from this thesis point of view the safest thing to do. This gives a more secure result and therefore a PE game gives, according to this, a safer

attitude seen from a security point of view. Therefore, PE games is used for the assessment of risk attitudes in this thesis.

2.4 The Utility Function

The utility function represents the risk attitude of a person. If two persons have different risk attitudes then they have different utility functions. This means that the utility function is individually determined.[7]

Determining the utility function is aided by the use of expected utility (EU) which is calculated for each game. To calculate the EU for the game presented in equation 2.1, the equation in 2.2 is used.

$$u(x_c) = p * u(x_1) + (1 - p) * u(x_2) \tag{2.2}$$

To be able to calculate the recursive equation of 2.2, the expected utility for each outcome x_1 and x_2 are needed. Let $\overline{x}$ be the highest outcome value for a series of games. Then the expected utility for $\overline{x}$ equals 1.0. Similarly, $\underline{x}$ denotes the lowest outcome value for the same series of games and the utility for $\underline{x}$ is set to 0.0. [7]. This way the utility for the game using the two outcomes $\overline{x}$ and $\underline{x}$, can be calculated.

To be able to calculate all utility values there are some restrictions about the possible outcomes x_1 and x_2 for any games in a series. For any game, the outcomes must be either $\overline{x}$, $\underline{x}$, or a x_c from another game.

To exemplify, calculation of a series of three games are shown in figure 2.2. These games have the different lottery outcomes x_1, x_2 and x_3. The probabilities p_1, p_2 and p_3 in figure 2.2 are set by the player, since PE games are used for determine the utility. Then expected utilities $u(x_3)$, $u(x_4)$ and $u(x_5)$ are calculated as in figure 2.3. The lowest of all outcome values is x_2 and the highest is x_1. Together all expected utilities result in the utility function u(x).

When playing games the player is assumed to be consistent in its decisions. According

$$x_3 \sim \begin{cases} (p_1, x_1) \\ (1-p_1, x_2) \end{cases}, \ and\ x_1 > x_3 > x_2$$
$$x_4 \sim \begin{cases} (p_2, x_1) \\ (1-p_2, x_3) \end{cases}, \ and\ x_1 > x_4 > x_3$$
$$x_5 \sim \begin{cases} (p_3, x_3) \\ (1-p_3, x_2) \end{cases}, \ and\ x_3 > x_5 > x_2$$

Figure 2.2: Relationship between games

$$\begin{aligned} u(x_1) &= 1 \\ u(x_2) &= 0 \\ u(x_3) &= p_1 * u(x_1) + (1-p_1) * u(x_2) \\ u(x_4) &= p_2 * u(x_1) + (1-p_2) * u(x_3) \\ u(x_5) &= p_3 * u(x_3) + (1-p_3) * u(x_2) \end{aligned}$$

Figure 2.3: Utility calculations

to French [7], a player is consistent when the player always prefers a higher monetary value to a lower monetary value. See equation 2.3.

$$x_i \geq x_j \Leftrightarrow u(x_i) \geq u(x_j) \text{ for any } x_i, x_j \tag{2.3}$$

In figures 2.4 - 2.7, some examples of graphs drawn from utility functions are given. In the graphs, C is the value for certain payoff and U is the utility for that payoff. In figure 2.4 a utility function for a risk averse person is shown. As can be seen, the curve for a risk averse person is concave. In figure 2.5 the person's risk attitude is risk prone. As can be seen, the curve for a risk averse person is convex. In figure 2.6 the person's risk attitude is risk neutral, and therefore is linear. In figure 2.7 the person's risk attitude changes from risk averse (A) to risk neutral (N) and from risk neutral to risk prone (P).

The range of the utility function $u(x)$ is $0 \leq u(x) \leq 1$. Its domain is defined by the interval $[x_1, x_2 \ldots x_n]$, and the function is strictly increasing on that interval. The utility function $u(x)$ can be seen as an ordered set of values $S = \{(x_1, u(x_1)), (x_2, u(x_2)), \ldots, (x_n, u(x_n))\}$.

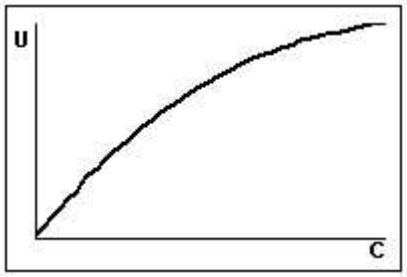

Figure 2.4: Risk averseness

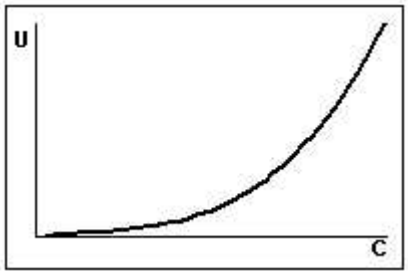

Figure 2.5: Risk proneness

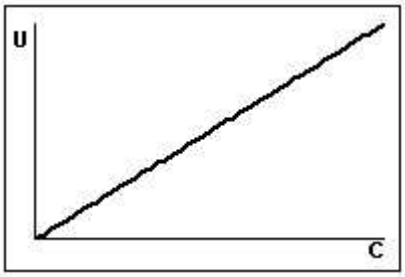

Figure 2.6: Risk neutrality

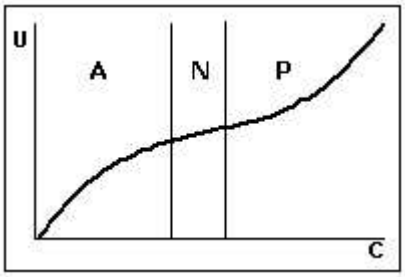

Figure 2.7: Risk averse, risk neutral and risk prone

To simplify the usage of the utility function in an application, a mathematical model of the utility function is needed.

In this thesis the model is assumed, based on figure 2.7, to be a polynomial function of the third order, see section 3.3. This assumption is reasonable, because most DMs follow that behaviour. The model is fitted using regression analysis on the set of values S. Although the resulting function $U(x)$, is mathematically valid in the interval $[-\infty, +\infty]$, the approximation is only valid in the interval $I = x_{min} \leq x \leq x_{max}$. The range of the function $U(x)$ does not necessarily have the same range $0 \leq U(x) \leq 1, \forall x \in I$, as $u(x)$, because the approximation of $U(x)$ takes equally care of each value. Sometimes the approximation, will even be invalid in regards to the strictly increasing property, imposed on the function $u(x)$, as the regression does not care about such restrictions.[1]

Since the domain of the new function $U(x)$ is I, any point $x \in I$ results in a valid function value no further interpolations are needed. After all computations are made, the

persons risk attitude is specified by the four coefficients of a polynomial.

2.4.1 Risk Premium

According to French [7] a **risk premium** "is the maximum part of the expected monetary value that the decision maker is prepared to forfeit in order to avoid the risk associated with a lottery."

The expected monetary value (EMV) is calculated as shown in equation 2.4. The EMV is the average payoff of a lottery if played many times (a finite number of times) [7]. Again, the game presented in equation 2.1 is used as model for the theories.

$$EMV = p * x_1 + (1 - p) * x_2 \tag{2.4}$$

With the help of expected monetary value, the risk premium can be calculated using equation 2.5. x_c is the certain payoff.

$$\pi = EMV - x_c \tag{2.5}$$

The sign of π indicates a person's attitude to risks. Thus,

$$\begin{aligned} \pi &> 0 \text{ indicates risk averseness.} \\ \pi &= 0 \text{ indicates risk neutrality.} \\ \pi &< 0 \text{ indicates risk proneness.} \end{aligned}$$

Consider an example where two persons are offered the same PE-game as in figure 2.8.

$$500 \sim \begin{cases} (p, 1000) \\ (1 - p, 0) \end{cases}$$

Figure 2.8: The PE-game in the risk premium example

Person (A) sets the probability p to a value that makes it indifferent between getting the 500 or playing the lottery. Maybe to a value of 0.4. Then $EMV = 0.4 \cdot 1000 + 0.6 \cdot 0 = 400$, which makes $\pi = 400 - 500 = -100$. Person (B) is playing the same game but sets the p to a value of 0.6, then $EMV = 0.6 \cdot 1000 + 0.4 \cdot 0 = 600$, which makes $\pi = 600 - 500 = 100$. This makes person A risk prone, whereas person B is risk averse. To be indifferent between getting 500 or playing the lottery person A must be offered a lottery with an average payoff of 400. Person B must be offered a lottery with an average payoff of 600 to be indifferent. A risk neutral person would be indifferent when offered a lottery with an average payoff of 500, which gives a risk premium of 0.

Chapter 3

Regression Analysis

"There are three kinds of lies: lies, damned lies, and statistics."
– Benjamin Disraeli, British politician (1804 - 1881)

3.1 Introduction

This chapter describes how the utility function from the previous chapter is approximated to a mathematical function. This is done by regression analysis, using the method of least squares. It is also shown how to adapt the least square method to derive third order polynomial as required for risk attitude functions. Some statistical tests are also described, like the confidence interval and R^2-test, which are used to check how good the approximation represents the actual values.

3.2 Regression Analysis

The purpose of single variate regression analysis is to adapt a mathematical model to a set of values. This is useful, for instance when trying to determine a relation between data. In figure 3.1, a straight line has been approximated to the values represented by dots. One

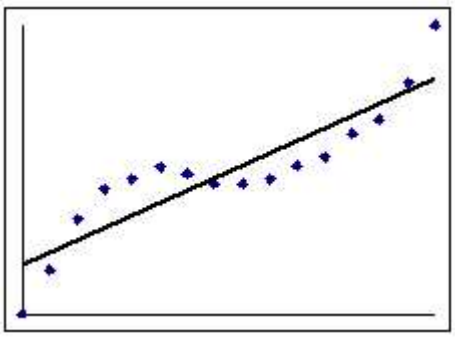

Figure 3.1: A linear approximation.

way of finding the approximation is by using the method of least squares. Equation 3.1 shows the formula for multi variate regression, for n variables. Multi variate regression is used when the relation is based on multiple factors, that is the model is approximated to the set of $(y, (X_1, X_2, \ldots, X_n))$ values. [6]

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_n X_n \tag{3.1}$$

The idea behind the method of least squares is to find the coefficients $\beta_0, \beta_1, \ldots, \beta_n$ that make the sum Q in equation 3.2 as small as possible. Equation 3.2 is the sum of all squared differences from an approximated value $\hat{y}_i$ to the y_i value.

$$Q = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i} - \beta_3 X_{3i} - \ldots - \beta_n X_{ni})^2 \tag{3.2}$$

To establish the so called *normal-equations*, see equation 3.3, the partial derivative of the function Q with respect to each β_i is calculated.[6], which results in equations 3.3.

$$\begin{aligned}
&\tfrac{\partial Q}{\partial \beta_0} = -2\textstyle\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i} - \beta_3 X_{3i} - \ldots - \beta_n X_{ni}) \\
&\tfrac{\partial Q}{\partial \beta_1} = -2\textstyle\sum_{i=1}^{n} X_{1i}(y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i} - \beta_3 X_{3i} - \ldots - \beta_n X_{ni}) \\
&\vdots \\
&\tfrac{\partial Q}{\partial \beta_n} = -2\textstyle\sum_{i=1}^{n} X_{ni}(y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i} - \beta_3 X_{3i} - \ldots - \beta_n X_{ni})
\end{aligned} \tag{3.3}$$

Then all partial derivates, from equation 3.3, are set to 0 to make a linear equation system. After the equations are simplified the result is the equations in 3.4.

$$\begin{aligned}
&\textstyle\sum_{i=1}^{n} y_i - \beta_0 n - \beta_1 \sum_{i=1}^{n} X_{1i} - \ldots - \beta_n \sum_{i=1}^{n} X_{ni} = 0 \\
&\textstyle\sum_{i=1}^{n} y_i X_{1i} - \beta_0 \sum_{i=1}^{n} X_{1i} - \beta_1 \sum_{i=1}^{n} X_{1i}^2 - \ldots - \beta_n \sum_{i=1}^{n} X_{1i} X_{ni} = 0 \\
&\vdots \\
&\textstyle\sum_{i=1}^{n} y_i X_{ni} - \beta_0 \sum_{i=1}^{n} X_{ni} - \beta_1 \sum_{i=1}^{n} X_{ni} X_{1i} - \ldots - \beta_n \sum_{i=1}^{n} X_{ni}^2 = 0
\end{aligned} \tag{3.4}$$

The equations in 3.4 can be written as an augmented matrix $Ax = b$, resulting in the matrix in equation 3.5.[2]

$$A = \begin{bmatrix} n & \sum_{i=1}^{n} X_{1i} & \ldots & \sum_{i=1}^{n} X_{ni} \\ \sum_{i=1}^{n} X_{1i} & \sum_{i=1}^{n} X_{1i}^2 & \ldots & \sum_{i=1}^{n} X_{1i} X_{ni} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} X_{ni} & \sum_{i=1}^{n} X_{ni} X_{1i} & \ldots & \sum_{i=1}^{n} X_{ni}^2 \end{bmatrix}, x = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{n-1} \end{bmatrix}, b = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} y_i X_{1i} \\ \vdots \\ \sum_{i=1}^{n} y_i X_{ni} \end{bmatrix}$$

$$A|b = \left[\begin{array}{cccc|c} n & \sum_{i=1}^{n} X_{1i} & \ldots & \sum_{i=1}^{n} X_{ni} & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} X_{1i} & \sum_{i=1}^{n} X_{1i}^2 & \ldots & \sum_{i=1}^{n} X_{1i} X_{ni} & \sum_{i=1}^{n} y_i X_{1i} \\ \sum_{i=1}^{n} X_{2i} & \sum_{i=1}^{n} X_{2i} X_{1i} & \ldots & \sum_{i=1}^{n} X_{2i} X_{ni} & \sum_{i=1}^{n} y_i X_{2i} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sum_{i=1}^{n} X_{ni} & \sum_{i=1}^{n} X_{ni} X_{1i} & \ldots & \sum_{i=1}^{n} X_{ni}^2 & \sum_{i=1}^{n} y_i X_{ni} \end{array}\right] \tag{3.5}$$

The matrix in equation 3.5 is solved using $x = A^{-1}b$, assuming that A is invertible. A matrix A is invertible if there is a matrix B satisfying: $AB = BA = I$, where I is the

identity matrix[1].[2]

A matrix that is not invertible is called singular, which is the word used when talking about unsolvable matrices later on in the thesis.

3.3 Approximating a Third Order Polynomial

As presented in the previous chapter, the utility function looks like a polynomial of the third order. A third order polynomial is defined as in equation 3.6.

$$y(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 + \alpha_3 x^3 \tag{3.6}$$

The regression determines the coefficients: $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ of the polynomial in equation 3.6. This is done by using three variables in equation 3.1, and setting them accordingly to $X_1 = x, \quad X_2 = x^2, \quad X_3 = x^3$. Using the least square method gives as result the augmented matrix in figure 3.7, which when solved reveals the coefficients: $\alpha_0, \alpha_1, \alpha_2, \alpha_3$.

$$\left[\begin{array}{cccc|c} n & \sum_{i=1}^{n} X_{1i} & \sum_{i=1}^{n} X_{2i} & \sum_{i=1}^{n} X_{3i} & \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} X_{1i} & \sum_{i=1}^{n} X_{1i}^2 & \sum_{i=1}^{n} X_{1i}X_{2i} & \sum_{i=1}^{n} X_{1i}X_{3i} & \sum_{i=1}^{n} y_i X_{1i} \\ \sum_{i=1}^{n} X_{2i} & \sum_{i=1}^{n} X_{1i} * X_{2i} & \sum_{i=1}^{n} X_{2i}^2 & \sum_{i=1}^{n} X_{2i}X_{3i} & \sum_{i=1}^{n} y_i X_{2i} \\ \sum_{i=1}^{n} X_{3i} & \sum_{i=1}^{n} X_{1i}X_{3i} & \sum_{i=1}^{n} X_{2i}X_{3i} & \sum_{i=1}^{n} X_{3i}^2 & \sum_{i=1}^{n} y_i X_{3i} \end{array}\right] \tag{3.7}$$

Figure 3.2 shows an approximation of a third order polynomial, using the same data as in figure 3.1.

3.4 Checking the Approximation

Some measurements are needed to determine how well the model was approximated to the values. To do this, the term residual is defined, since it is used by the R^2 *test*.

[1] An identity matrix is a matrix where all entries are 0, except the diagonal entries which are 1.

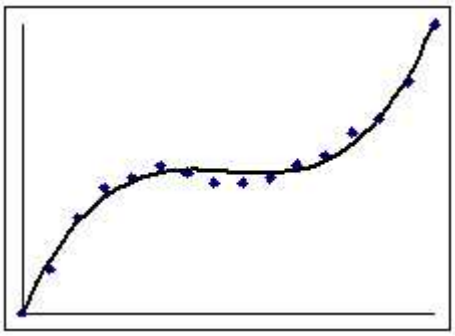

Figure 3.2: A third order polynomial approximation of data.

The difference between the approximated value $\hat{y}_i$ and the corresponding value in the set y_i is called the residual e_i and is calculated as shown in equation 3.8.

$$e_i = y_i - \hat{y}_i \tag{3.8}$$

3.4.1 R^2 Test

R^2 is a measurement indicating the success of a regression, see equation 3.9 for definition. According to Draper and Smith [6] the "R^2 value measures the 'proportion of total variation about the mean $\bar{Y}$ explained by the regression"'. It is a percentage explaining how much of the variations in data that can be explained by the model.

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.9}$$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y	0	2.11	4.55	6.06	6.57	7.16	6.8	6.32	6.32	6.57	7.2	7.6	8.76	9.43	11.2	14

Table 3.1: Example data

The data in table 3.1 was generated from a third order polynomial, chosen at random. Then some of the values were either increased or decreased making the date not to be an

exact polynomial. This data were then used to build the diagrams in figures 3.1 and 3.2. The R^2-values were calculated for each approximation (both the linear and the third order polynomial). Table 3.2 shows the result.

linear	0.7979
polynomial	0.9893

Table 3.2: R^2 test results.

The results from table 3.2 would not surprisingly indicate that the approximation of a third order polynomial is a better fit for that particular set of data.

3.4.2 Confidence Interval

Two measurements of the average, of a set of values, are the mean value and the median. If the set of values contains values that lie in a cluster, and one or two of the values lies far away from these, then there is a big difference between the mean value and the median. The median is not as sensitive as the mean value in the case of such values. The mean value is the sum of all values divided with the number of values. The mean value of n numbers $\{x_1, x_2, \ldots, x_n\}$ is given by equation 3.10.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{3.10}$$

The median value is chosen from a sorted set of numbers so that there is equally many numbers greater than and smaller than the median value. If there is an odd amount of values in the set, then the value that lies in the middle of the set is the median. If the set contains an even amount of values, then the average of the two middle-values is the median for that set. For instance, in a sorted set of five values the third value would be the median for that set.

The confidence interval is an interval with a grade called confidence grade connected

to it, as the interval without this grade is not saying anything. The confidence grade is a measurement of the probability that, if sampling many times, the median is lying inside the boundaries of the interval. If sampling 100 times then a confidence grade of 95% means that the median is lying inside the interval in 95 of the cases and in 5 of the cases it lies outside the interval. So, if the sampling should continue, one could with 95% certainty predict that the interval would contain the median.

When deciding the confidence interval the values of the samples are first sorted in an ascending order. Then the values are numbered from 1 to n. The largest interval that can be is $[1, n]$, the next largest is $[2, n-1]$ and so on. Depending on how great the confidence grade should be, the interval is narrowed so the grade is close to the desired grade. The desired grade and the calculated grade can almost never be the same. Therefore a calculated grade that is greater than or equal to the desired one is chosen. To calculate the confidence grade for the interval the formula in equation 3.11 is used.[13]

$$grade = 1 - 2 \cdot \sum_{i=0}^{k} \left(\left(\begin{array}{c} n \\ i \end{array} \right) \cdot 0.5^{i} \cdot 0.5^{n-i} \right) \tag{3.11}$$

In equation 3.11, n is the total amount of values in the set, k is the narrowing factor and n *over* i is defined as in equation 3.12.

$$\left(\begin{array}{c} n \\ i \end{array} \right) = \frac{n!}{n! \cdot (n-i)!} \tag{3.12}$$

With $n = 8$ in equation 3.11, the interval $[1, 8]$ has $k = 0$ and the interval $[2, 7]$ has $k = 1$ and so on. When the interval is narrowed the grade is decreased. The greater amount of values in the set, the greater the confidence interval can be at the same time as a narrow interval.

The set of values can for instance be the distance between the actual data and the regressed values. Then the median is telling the average distance between the regressed

and the actual values. The confidence grade is the probability that the next distance would fall into the confidence interval.

Chapter 4

Development Approach

"Good judgement comes from experience, and experience comes from bad judgement."
– Barry LaPatner

4.1 Introduction

The theories needed for determining risk attitudes has now been investigated and it is time for the next part of the thesis to start. From now on the focus will be on the development of the application to utilize these theories.

This chapter first introduces Extreme Programming (XP), which is the software development method used in this project. Then the development environment for this project is described. Finally, a section describing the need and motivation for prototyping and testing follows.

4.2 Extreme Programming

XP is an lightweight agile method for software development, put forward by Kent Beck [3]. XP includes many practices, but in this project only a few were used since only two people were involved in it.

In XP, a **metaphor** is selected for a project when it begins. A metaphor is a simple, shared story of how the whole system works, and that guides all development through the project. The metaphor for this project was: Build an application to assess risk attitudes.

Pair programming is one of the practices in XP. The idea is to have two people sitting together working on one computer. One person codes, and the other person reviews the code and helps with design issues. Every now and then they switch roles, so the coder becomes the reviewer, and vice versa. The continuous reviewing of new code minimizes coding errors.[3]

According to XP, **testing** should be performed early and continuously during development to ensure code compatibility. Preferably, a test should be implemented prior to coding. Code is then written to make the test run successfully. The JUnit framework, see section 4.5.2, has been used for testing in this project, as it supports automatic testing in an XP project.

In XP, development proceeds according to **stories and tasks**. A story consists of requirements about the functionality of the application. A story has a priority, set by the customer, stating how important that particular story or functionality is. From each story the development team then creates tasks. Each task is implemented by a pair of developers.

In an XP project, the code should be written as **simple** as possible, as long as it solves the current task. Simple code is easier to understand and contains less faults [3]. Continuous **refactoring** is another important practice in XP. Refactoring means that the code-design is restructured without changing the code's behaviour. Usually refactoring moves from one design to another, as the need for a more fitting design becomes evident,

due to increased code size. In XP, the code design should be modified whenever it is suitable to do so, to ensure that the simplest and best design is always used.

In XP, every **release** is planned with a **planning game**, where the stories are prioritised and the content of the release is roughly estimated [3]. In this project the releases were planned in the start of each **iteration**.

The XP practice of **continuous integration** says that the project should be built and integrated after each task is completed.[3] This step was not needed in this project because of the size of the project and the number of developers.

4.3 The Development Environment

The NetBeans IDE[1] was chosen because the authors were familiar with that IDE. Since the selection of an IDE does not reflect the resulting code, no further investigation of other IDEs seemed necessary. NetBeans has built-in support for user interface design and JUnit[2] tests. This helped to fulfill the testing practices of XP.

It was decided that the Concurrent Versions System (CVS) was to be used because one of the authors already had a CVS server up and running. The author's prior knowledge of CVS ensured that it would be preferable to other version control systems.

CVS centralises all code and other files in a repository stored on a server. This allows for a distributed development model by letting each client use its own subset of the CVS repository. Each client then updates the repository with only their modifications. If a conflict should occur, that is when two clients want to make similar updates to the same files, then the CVS will stop the update of the second client until that client has resolved the conflict. The centralised code base makes backups, builds easier since all code is located in one place.

[1] An IDE or Integrated Development Environment, is an application that aids in writing code, user interface design, compilation and debugging of software.

[2] JUnit is a framework for automated tests. See section 4.5.2

4.4 Prototyping

According to Sommerville [11] "The principal purpose with prototyping is to validate software requirements". This means that prototypes are useful to demonstrate that the requirements are captured and understood. The two types of prototypes that are used in this thesis are called throw-away and evolutionary prototypes.

A throw-away prototype is used to clarify the requirements of an application. Although a throw-away prototype should not be used in further development, parts of the prototype can later be used in production code.[11]

An evolutionary prototype, starts with an incomplete application and as requirements become clear more functionality is added. This way the prototype evolves to an application with all the required functionality.[11]

4.5 Testing

According to Binder [4], "software testing is the execution of code using combinations of input and state selected to find bugs".

A **fault** is defined as missing or incorrect code. When a block of code that contains faulty code is executed, a **failure** might occur. A failure is code that during its execution does not do what it is supposed to. This can manifest in a crash of the program or as an erroneous output. Faults are created as a developer makes coding errors, or by bad hardware.[4]

Behavioural testing or blackbox testing, are tests based on an external view of a component.[4] The term component is used for the entity (method, class or system) currently being tested.

A **testcase** specifies the component's state before and after the test, and performs the actual test. If the actual result would differ from the expected result, then the test has failed. Test cases can be grouped into one or more **testsuites**. [4]

4.5.1 Test Coverage

The following method *int sum(int a, int b)*, which returns the sum of two parameters a and b, serves as an example of a test case. One of many test cases for the method is the assertion that *sum(20,30) = 50*. This only validates that the method call is valid for the specified arguments of the test case. If for instance *sum(21,31)* also would result in 50, the assertion that *sum(21,31) = 52* would fail.

The example above demonstrates that a component does not need to be free of faults after passing one test case. In fact, there is no guarantee that a component is fault free, even after several test cases with different arguments has successfully been run. [4]

However, the set of all possible test cases, called an exhaustive test suite, for the method *int sum*(*int a*, *int b*) is quite big since it consists of all possible combinations of a and b. If *int* is a 32-bit datatype, then there are 2^{64} different combinations for the values of a and b, and equally many test cases. This is too many for most cases, or as Binder says regarding exhaustive test cases: "Software testing is therefore necessarily concerned with small subsets of the exhaustive test suite.". [4]

4.5.2 The JUnit Framework

JUnit is a unit testing framework for Java. It facilitates automated testing using both test cases and test suites. JUnit is written by Erich Gamma and Kent Beck. Further information about JUnit can be found at *http://www.junit.org*. JUnit was used throughout the development to create unit tests for the source code.

4.5.3 The jfcUnit Framework

The project owner proposed that jfcUnit should be used to test user interface specific code. jfcUnit is a framework that allows testing of Swing-based graphical user interfaces. jfcUnit is based on JUnit, and handles test cases and test suites the same way. With jfcUnit tests

that asserts certain application responses to user interaction, such as menu selections or the press of a specific button, can be created. Further information about jfcUnit, can be found at *http://jfcunit.sourceforge.net/*.

Chapter 5

The Prototypes

"Get your facts first, and then you can distort them as much as you please."
– Mark Twain (1835-1910)

5.1 Introduction

The project owner wanted two prototypes developed, prior to the Mmv application. The first prototype presents a simple application exemplifying a user interface (UI). Being able to perform regression analysis was considered a critical functionality of the Mmv application. Thus, the purpose of the second prototype was to find a way of performing regression analysis.

Throughout this chapter the design decisions taken while developing the prototypes are explained, as well as comments regarding each prototype when presented to the project owner.

5.2 Prototype Planning

Each prototype was planned with the project owner, and resulted in six stories for the first prototype, and two stories for the second prototype. See appendix A.1 and A.2, respectively.

The user interface prototype was developed as a throw-away prototype. This means that code is thrown away when the requirements are validated. The regression analysis prototype was an evolutionary prototype and was used as the regression analysis code in the Mmv application.

5.3 The First Prototype - User Interface

The purpose of the first prototype was to present a user interface for the project owner. The UI consisted of a menubar, a statusbar and three small windows. The options in the menubar had no functionality, except for the 'About'-option. When the user choose the 'About'-option a dialog appears on the screen with information about the prototype, such as version and copyright notice. The small windows were all empty, and had the purpose of showing that the mainframe can hold windows where information, such as game information or a table, can be shown. When the user tries to close the prototype a dialog appears on the screen, asking the user if it wants to save data before closing the prototype.

5.3.1 Design Decisions

At this level in the project, the decision about which file format that should be used for saving data regarding games, was thought to be a too early decision to take. This, and to remember the last file path were not implemented in the prototype, so the task 'Open File' in story 3 was abandoned. Since each story was rather small, no actual tasks were

needed for implementing them.

The prototype was a so called multi document interface (MDI). This means that the UI consists of a mainframe that may contain many windows, containing any type of content such as a diagram or a table with values. These windows are freely positionable and resizeable inside the application window. A MDI approach was decided to be used because multiple views of the same data were supposed to be used. All views are kept into the same application window and not spread all over the screen, which is the alternative if using a single document interface (SDI).

5.3.2 Results

After showing the prototype to the project owner some issues arose. The biggest concern was that the UI was not consistent in the graphical design. The 'About'-dialog did not have the right look and feel according to the 'Windows Look and Feel'-standard. This was remedied in the development of the Mmv application.

5.4 The Second Prototype - Regression Analysis

The first story of this prototype allows a user to input data into the prototype. The second story would approximate a third order polynomial to the data inputted in the first story using regression analysis.

The first story was implemented as a window with a table where the user could input a number of values. The implementation of the second story was rather straightforward, simply an implementation of the theories from section 3.3. A problem was encountered since Java does not contain a matrix class, needed to facilitate the final step of the regression analysis, namely solving the matrix in equation 3.7. Since developing a matrix class for this project would have taken too much time, the use of an external library was needed. The Jama library, see section 6.3, supported all functionality needed to solve the augmented

x	y		x	y
1.0	13.6		9.0	4360.0
2.0	60.6		10.0	5959.0
3.0	179.8		11.0	7908.6
4.0	406.0		12.0	10243.6
5.0	774.0		13.0	12998.8
6.0	1318.6		14.0	16209.0
7.0	2074.6		15.0	19909.0
8.0	3076.8			

Table 5.1: Test values used as input to regressions

matrix and was selected for this prototype.

5.4.1 Testing The Regression Analysis

The regression analysis was tested using Microsoft Excel as a reference. The regression in Excel is assumed to be correct. The test values in table 5.1 were used in both the prototype and Excel. A third order polynomial was approximated to the values in table 5.1 using regression analysis in each application. The results from the regression analysis is given in equation 5.1, where the coefficients values were rounded off to two decimals. Both Excel and the prototype gave the same end result.

$$y(x) = 5.80x^3 + 1.30x^2 + 2.50x + 4.00 \tag{5.1}$$

5.4.2 Results

The result of the second prototype is that the implemented module to perform regression analysis seems to be working properly. When testing the regression module, with data from table 5.1 as well as several other tests with varying data used as input, the approximation obtained was either similar or exact to the approximation from Excel. The eventual differences can be explained by roundoffs.

Chapter 6

The Mmv Application

"In these matters the only certainty is that nothing is certain."
– Pliny the Elder (23 AD - 79 AD)

6.1 Introduction

After the prototypes had been developed and presented to the project owner, the development of the Mmv application was started. This chapter describes the release plan and the external libraries used in the application. The Extensible Markup Language (XML), will also be briefly described in this chapter since some of the stories are depending on XML.

In the three following chapters the design and implementation of each iteration of the application is described.

6.2 The Release Plan

The Mmv application was developed with short release cycles according to XP. Each iteration was set to be two weeks long, and at the end of each iteration a release was shown to the project owner.

A total of four iterations was originally planned for the project, where each iteration was based on one or more stories that can be seen in appendix A.3. Each story was divided into smaller tasks. Table 6.1 shows the stories planned for each iteration.

First release:	Game Menu structure User interface Regression
Second release:	Save game Game settings
Third release:	Load game Export and Import
Fourth release:	Statistical Tests

Table 6.1: The initial iteration plan

The iteration plan was revised after the second iteration, see table 6.2. As mentioned in section 9.2, the fourth iteration was cancelled. The 'Load games' story had already been implemented in the second iteration and there was a decision that the priority for the 'Import and Export' story, originally destined for the third iteration, should be lowered. Due to this, the statistics story was moved to the third iteration, so there were no stories left for a fourth iteration. In addition, the time for the project was almost used and further stories considered outside the scope.

6.3 External Libraries

The need for a library to solve matrix systems became evident in the second prototype, see section 5.4. The prototype used the Jama Matrix API. The Jama API can be found

First release:	Game Menu structure User interface Regression
Second release:	Save game Game settings Load Game
Third release:	Statistical Tests

Table 6.2: The revised iteration plan

at *http://math.nist.gov/javanumerics/jama*. According to the Jama web site, the API will be proposed as a Java standard for general matrix handling. Since Jama proved to work well in the prototype, it was decided to be used in the Mmv application as well.

The stories in the initial iteration plan, see table 6.1, showed the need for a way to handle XML files. XML is explained in section 6.3.1. Since the authors had prior knowledge of the JDOM XML API, that API was chosen without further investigation, knowing that all requirements from the stories would be met. JDOM is a Java based document object model API for XML files. It allows for creation, loading and saving of XML documents, see *http://www.jdom.org* for more information.

6.3.1 XML

The Extensible Markup Language (XML) is a markup language based on the Standard Generalized Markup Language (SGML). XML focuses on the structure and contents of a document, instead of how it should be represented. An XML document contains a hierarchy of elements. Each element can contain attributes, information or child elements.[14]

To be considered well-formed, an XML document must obey the following rules:

- An XML document must contain only one root-element. A root-element is the element that contains all other elements.
- All elements must be properly nested.

- All attributes must be quoted.
- All end tags are required.

A well-formed document which also comforts to a DTD, is called a valid document.[12]

A DTD, or Document Type Definition, defines the structure, or language of an XML document. The DTD describes the relationship between elements, which elements that are allowed as children of another element, which elements that are mandatory and the multiplicity of elements. A DTD also defines which attributes that are valid for each element.[12] Since a DTD provides sufficient control over the structure needed in this prototype, the more advanced XML-schema was not needed. ’

Chapter 7

The First Iteration

"Doubt is not a pleasant condition, but certainty is absurd."
– Voltaire (1694-1778)

7.1 Introduction

This chapter describes the first iteration of the Mmv application. Together with the added functionality of playing games like in figure 2.1, this iteration implemented the stories as planned for the first iteration, see section 6.2. Some problems related to the regression analysis are described in section 7.3.2.

7.2 Story and Task planning

According to the plan, see section 6.2, the stories to implement in this iteration were: 'Game', 'User interface', 'Menu structure' and 'Regression'.

7.3 Implementation

The regression analysis component from the second prototype was used in this iteration.

7.3.1 Game Structure

As a design decision it was decided that all games would be stored in a list and created using recursion. The main reason for this decision was that it was quite easy to build the game structure in a recursive way, as the game structure must resemble the recursiveness of the utility function, see section 2.4.

Figure 7.1 shows the dependencies between the games, not the storage structure. In the figure each game is represented by a box, where O1 and O2 are the outcomes and E is the estimate for each game. The outcome values for the first game are set to 10000 and 0 as

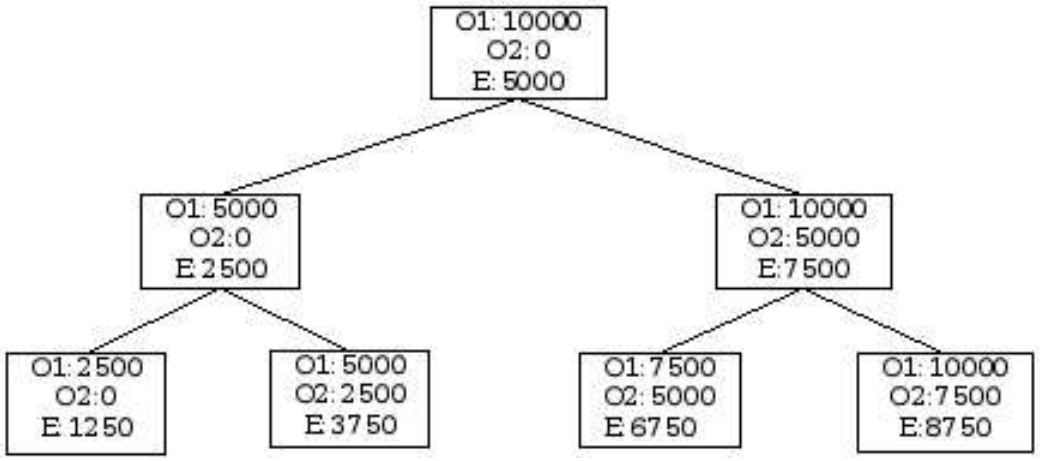

Figure 7.1: Example of the implicit game tree structure

specified in the story. The estimate is always the mean value of the two outcomes. Then, two more games were built based on this first game. One of these two got its outcomes based on the low outcome and the estimate from the first game. The other game of these two got its outcomes based on the high outcome and the estimate from the first game.

Then the procedure continued in a recursive manner until the desired amount of games was created.

The game structure was chosen since the utility calculations for each estimate E needs the utility of the outcomes $O1$ and $O2$, as described in section 2.4.

7.3.2 Regression

The regression analysis component from the second prototype was used in this iteration. Before it could be used, JUnit tests had to be created for the component.

Despite the successful JUnit tests, it turned out that the regression analysis had some problems. Since it is not possible to solve the matrix in equation 3.7 if it is singular, the regression analysis will fail for some combinations of the utility values if used as input.

There is no easy way to determine which games or utility values that will result in a singular matrix. Therefore the implementation was to always try to analyse, and if it failed take care of that failure.

The data used in the analysis are the calculated utility $u(estimate)$ and the estimate for each game. This leads to a polynomial representation of the user's utility function, or $U()$ as mentioned in section 2.4

7.4 The First Release

The first iteration contained the menu structure, a game panel, regression analysis and a text display showing the results of the regression analysis. The only options in the menu that were functioning were on the 'File'-menu and the 'Help'-menu. On the 'File'-menu, only the options 'New session' and 'Exit' were functioning and on the 'Help'-menu only the 'About' option was functioning.

In the following text, values in parenthesis are the values shown in the figure 7.2. The figure is a representation of the PE-games described in figure 2.1. The textfield (45) was

an editable field where the probability for outcome one (10000) should be entered. Then there were text labels for the certain outcome (5000), outcome one (10000), outcome 2 (0), and also the probability for outcome two (55). When the probability for outcome one was altered the probability for outcome two was also altered. There were also two buttons, one for advancing to the next game and one for stopping the game suite.

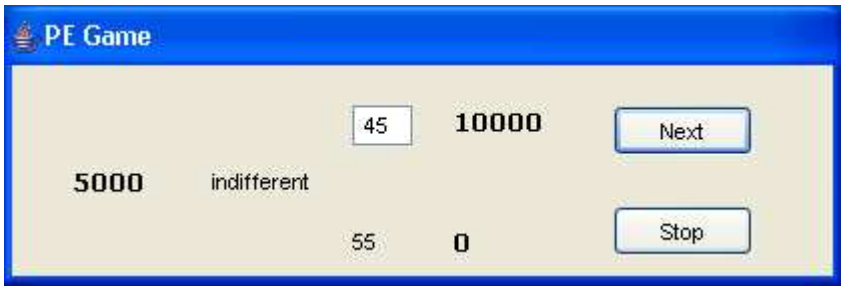

Figure 7.2: The game window for the first release.

After the 20 games were played a dialog was displayed on the screen, informing the user that all games had been played. When the dialog was confirmed the utility function was determined by first calculating the utility for each game. Then regression analysis was started and a text display was opened showing the utility function.

7.4.1 Comments

When the first release was presented, the project owner had a few comments and change requests. Some of the changes were related to the design of the window used for playing games. The symbol '%' should be put to the right of the two probability fields. An average mean value should be shown between the two outcomes. Two additional buttons were desired by the project owner. One of theses buttons has the same functionality as the 'Enter' key in the probability field. This button should have the caption 'Ok'. Then a second button, for clearing the probability field, was desired. This button should have the caption 'Clear'. The caption of the stop-button should be changed to 'Cancel' instead.

The locations of the buttons should be changed for easier usage. Then there should be a symbol for indifference above the 'indifferent' text.

The project owner also had a few suggestions about the logic of the game window. The field for inputting the probability value is desired to be a spinner. A spinner is a field where the value can be either increased or decreased with two buttons. There should be a possibility to enter the probability value in two different manners. Either as a decimal between 0 and 1, or as an integer between 0 and 100. This is motivated as some users prefer one notation of probability over the other. The error message that is shown when a user enters an illegal value should also be changed due to this fact.

There was also a desire for higher outcome values, due to the fact that the difference between the two outcomes after a few games became too small. Small differences between the outcomes makes the player indifferent and the game is of no use. A more randomly distribution of the games in a suit was also desired.

Chapter 8

The Second Iteration

"A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."
– Douglas Noel Adams, British author, Hitchhiker's Guide to the Galaxy (1952-2001)

8.1 Introduction

In this chapter the continued development of the Mmv application is described. The internal game structure was remade, saving files was implemented, and the UI of the game window was modified according to the requests made by the project owner.

8.2 Story and Task planning

The two stories 'Save game' and 'Game settings' were originally planned for this iteration, see section 6.2. But after the first release, more stories and tasks were defined, to accommodate the change requests made by the project owner. Most of these changes concerned

the look of the game window.

8.3 Implementation

Most of the development done during the second iteration concerned new functionality. Loading and saving games were implemented. As XP says that the design would be refactored whenever suited, a new application model solving some design issues was created. Also the way games were created and stored internally was modified, to accommodate the change requests made by the project owner.

8.3.1 New games dialog

When the user selects 'New Session' from the menu a dialog window is shown, asking for the maximum and minimum outcome values as well as the total number of games to create for the game suite. Some constraints were set on the input values, so the game creation procedure would create games from valid data.

8.3.2 Modifications to The Game Window

The suggestion from the project owner to use a spinner control to select the probability value was difficult to implement, because there are two different types of representation of the probability value, namely decimal and integer values. This called for a change of priority that resulted in that the spinner was postponed. More changes to the game window are described in section 8.4.

8.3.3 Game Structure

The game structure in the first iteration, see section 7.3.1, stored games internally in a linked list. The list was populated as the method for game creation recursively generated

the game structure. The way the recursion was devised, resulted in a list containing the in-order traversal of the recursion tree. Table 8.1 shows the result of an in-order traversal of the tree in figure 8.1.

This worked as all games were played in an order from the first game to the last game in the list. But as suggested by the project owner, the games seemed to be following a pattern, which they also did. The project owner wanted a more random distribution of the games during game play.

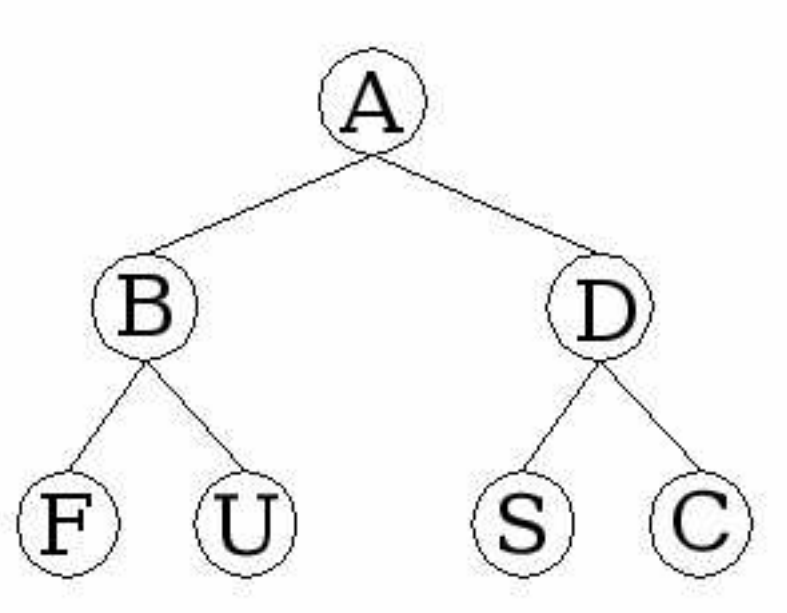

Figure 8.1: Example tree

It was decided that a breadth-first search of the game tree would result in a better distribution of the games to play. But, the linked list of games did not offer any way of being iterated in a breadth first way.

Then, the idea of using a binary heap came up. Since the structure of a binary heap represents an implicit tree structure no need for a recursive creation process would be necessary [15]. To create a new game, it is added to the next position of the heap, and one can then determine which values to use from the items already in the heap, using the

properties of the binary heap [15].

In the binary heap, games are stored in a breadth first search order, making the traversal as trivial as iterating from the first game to the last. Table 8.2 shows the traversal of the tree in figure 8.1. Notice that the first position of the table is not used since the first object in a binary heap is located at the second position. [15]

F	B	U	A	S	D	C

Table 8.1: In-order traversal

	A	B	D	F	U	S	C

Table 8.2: Breadth first traversal

The new solution for the game structure had more benefits. Additional games can easily be added to existing ones, simply by adding a new game to the heap. This was not possible before, as it would have demanded that the recursive calls all have been made again.

Also, even though it was not implemented, a method could be used to randomly select any game from the current level. This would have given an even better distribution of games, and the application would still be able to do regression analysis after each played game. This benefit would have been virtually impossible to do using the previous method.

8.3.4 Refactoring

To better accommodate the new functionality of the stories for the second iteration, the object oriented design had to be refactored, see section 4.2.

The refactoring meant that an application model class was created. The model now contains and defines all application logic. Some logic concerning games is still handled by the game handler as before. This distinction made it possible not to change so much of the code as a removal of the game handler would have caused.

The application model and user interface are separated and only communicates using the observer pattern. Using the observer pattern, the application model now notifies the UI (or whoever has registered for notifications) whenever the state of the application has changed. The UI is then responsible for taking appropriate actions, corresponding to the new state.

The observer pattern, defines a framework which allows several listeners to register for notifications from a subject. One of the benefits from the observer pattern is that coupling is loosened [9].

Like the rest of the UI, the analysis window also retrieves its data from the model when it is notified by the model.

8.3.5 Saving Games

In the story 'Save game' it is defined that games would be saved using XML, see 6.3.1. It was also decided that a DTD, see 6.3.1, was needed to design the format of the XML document.

The data needed to be saved for the game suite was the maximum and minimum outcomes. For each game the estimate, the probability, the two outcomes and a calculated utility for that game were also saved. To be able to determine whether a game had already been played or not, a boolean field called 'played' was added. The resulting DTD can be found in figure 8.2. An example of a save file, based on the DTD is shown in figure 8.3.

8.4 The Second Release

There were a few changes due to the comments done regarding release 1. First there were a few changes to the graphical design of the game window which can be seen in figure 8.4.

The '%' was added to the right of both the probability fields, to clarify that percentage

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT GameSuite (Game+)>
<!ATTLIST GameSuite>
<!ATTLIST GameSuite
max          CDATA      #REQUIRED
min          CDATA      #REQUIRED
nr           CDATA      #REQUIRED
>
<!ELEMENT Game EMPTY>
<!ATTLIST Game
 played (false | true ) #REQUIRED
 probability CDATA      #REQUIRED
 outcome1    CDATA      #REQUIRED
 outcome2    CDATA      #REQUIRED
 estimate    CDATA      #REQUIRED
 utility     CDATA      #REQUIRED
 >
```

Figure 8.2: The DTD used for save files

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GameSuite SYSTEM "gamesuite.dtd">
<GameSuite min="0" max="100" nr="10">
  <Game played="true" outcome1="100" outcome2="0" estimate="50" probability="0.5" utility="0.5" />
  <Game played="true" outcome1="50" outcome2="0" estimate="25" probability="0.4" utility="0.2" />
  <Game played="true" outcome1="100" outcome2="50" estimate="75" probability="0.34" utility="0.67" />
  <Game played="true" outcome1="25" outcome2="0" estimate="12" probability="0.64" utility="0.128" />
  <Game played="true" outcome1="50" outcome2="25" estimate="37" probability="0.34" utility="0.302" />
  <Game played="true" outcome1="75" outcome2="50" estimate="62" probability="0.34" utility="0.5578" />
  <Game played="true" outcome1="100" outcome2="75" estimate="87" probability="0.24" utility="0.7492" />
  <Game played="true" outcome1="12" outcome2="0" estimate="6" probability="0.34" utility="0.04352" />
  <Game played="false" outcome1="25" outcome2="12" estimate="18" probability="0.0" utility="0.0" />
  <Game played="false" outcome1="37" outcome2="25" estimate="31" probability="0.0" utility="0.0" />
</GameSuite>
```

Figure 8.3: A save file example

is the unit for the inputted value. The probability label was changed to a field. This was done to get a more uniform appearance. This field is not editable and therefore greyed. The average mean value was added to the game window right between the two outcomes. A button with the caption 'Clear' was added, with the functionality of clearing the input field. Also a button, with the same functionality as the 'Enter' key, was added to the game window. The former 'Stop' button has its caption changed to 'Cancel'. The location of all the buttons was altered for easier usage. The symbol for indifference was added above the text 'indifferent'. Also a few lines were added with the intention to clarify that it is a

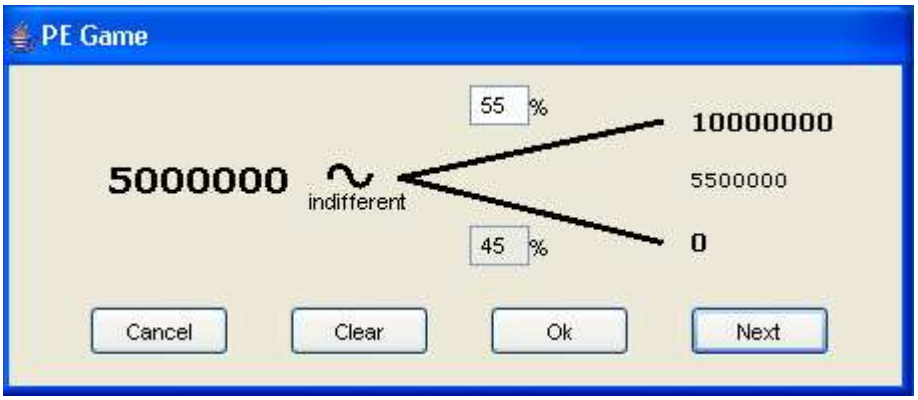

Figure 8.4: The game window for the second release.

lottery with two outcomes.

Then there were a few changes to the logic in the game window. Now it is possible to enter both decimal and integer values in the probability field. The range for the decimal is between 0 and 1, and the range for the integers is between 0 and 100. If the user enters an illegal value a dialog appears with a notification about this. Due to a suggestion from the project owner about higher outcome values, a dialog for setting the parameters for a game suite was developed for this release. The design of the dialog can be seen in figure 8.5.

In the dialog the minimum and maximum values for the outcomes in a game suite can be set. This way the user has the ability to set the values to what are tangible values for that user. There is also a field for setting the number of games to be played in a game suite.

8.4.1 Problems

Due to bad and insufficient tests, some problems occurred during the second release. A change in the logic that manages the check before closing the application, resulted in an application that was unclosable. If a test had existed prior to that modification, and the test had been run, then that error would have been found before showing it to the project

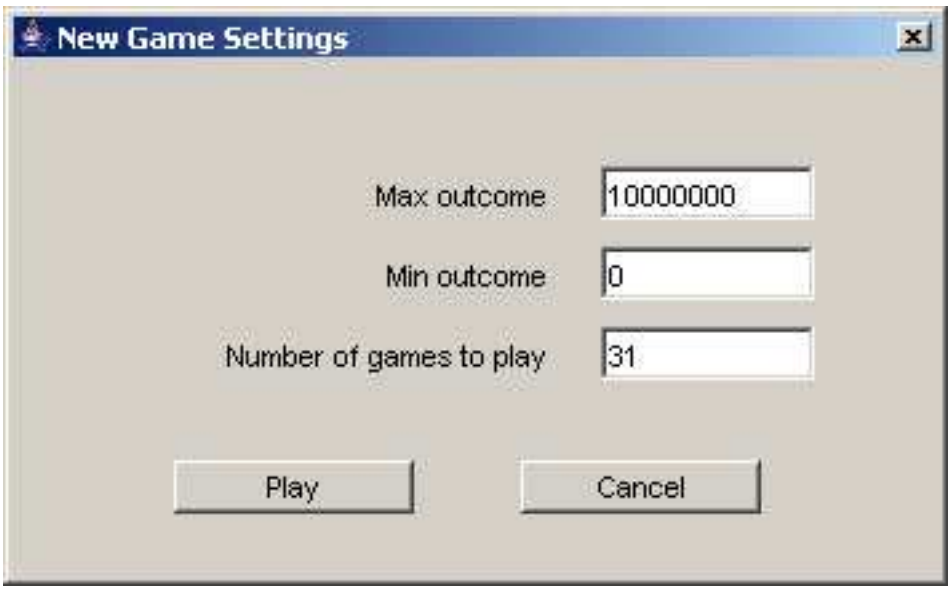

Figure 8.5: The dialog for game suite settings

owner.

8.4.2 Comments

The project owner had a few suggestions concerning the game window shown in the UI. The fields for the probabilities would have the width of four visible characters instead of three. To display the average mean value should be optional, so there is also the need of a check box for choosing if the value should be visible or not.

The buttons 'Ok' and 'Next' are desired to be one button with a double functionality. When there is something entered in the probability field then the button should have the same functionality as the former 'Ok' button. After the user has confirmed the inputted value, either by pressing the enter key or using the 'Ok' button, the button should have the same functionality as the former 'Next' button. When pressing the button in the latter mode the user can play the next game.

When entering values in decimal that are greater than 0.65 in the probability field,

the value shown in the probability field for outcome 2 showed only the last digits of the number, as the field was right justified. The fields must be left-justified to show just the first digits instead.

The project owner had a suggestion of a file extension, '.mmv' for the files that are saved from the application. As game data are stored using XML there is a DTD connected to each file. This DTD must reside in a specific catalog relative to the save file, for the loader to be able to validate the save file.

When the option 'Analyse' in menu 'Data Handling' was chosen the display with results from the regression analyser came up. If this was done twice, without first closing the display, another window came up with the same information as the first one. This is a faulty behaviour which should be fixed in the next release.

The project owner decided that the priority for the story 'Import and Export data', which was to come in the following iteration, was lowered. This decision was based on the fact that the save file contained enough data to be used in other applications.

Chapter 9

The Third Iteration

"Many of life's failures are people who did not realize how close they were to success when they gave up."
– Thomas A. Edison (1847 - 1931)

9.1 Introduction

In this chapter the continued development of the Mmv application is described. The user interface of the game window was modified according to the requests made by the project owner.

9.2 Story and Task planning

According to the revised iteration plan in section 6.2, the story to develop in this iteration was the 'Statistical tests'. A statistical test gives an indication to how good the regression analysis fitted the utility function to the game data.

Due to the comments from the project owner in the second release, a new story with tasks was created. This story and its tasks can be seen in Appendix A.4.

9.3 Implementation

The stories and tasks for this iteration resulted in changes of the game window and the development of statistical tests. These are described below in one section each.

9.3.1 Game Window

The tasks for the game window resulted in the new game window which can be seen in figure 9.3. As the visibility of the average mean value would be optional and chosen by the user, a check box was added to the game window. With this check box the user can choose whether to show or hide the average mean value during game play. The former buttons 'Ok' and 'Next' were removed from the game window that was developed in the second release. Instead another button with a double functionality was added. The two functionalities for the new button are the same functionalities as the former 'Ok' button and the former 'Next' button had. If the user has confirmed the entered probability value, either through pressing the 'Enter' key in the field or pressing the 'Ok' button, the new button has the same functionality as the former 'Next' button. If the user has not confirmed the entered probability value, the button has the same functionality as the former 'Ok' button.

The 'Clear' button was moved further away from the 'Cancel' button. This way it made a group together with the 'Next/Ok' button and the game window became more clear. During play of a game suite the only buttons that would be used are the 'Clear' button and the 'Next/Ok' button. As the 'Cancel' button is not grouped with the other buttons, this got more obvious to the player.

9.3.2 Statistical tests

Statistical tests were developed according to the story for this release. The statistical tests that were developed for this release were a R^2-test and a test for confidence interval with a confidence grade. The R^2-test and confidence interval test are described in section 3.4. The intention with these tests is to give the player an indication of when the regression analysis is good enough. If the result is good enough the game suite can be stopped and the results can be saved. Three results for the confidence interval were presented in the analysis window. Three intervals, one for each one of the confidence grades 95%, 97% and 99%, were printed out to give the player the opportunity to decide at which interval it is satisfied with the regression analysis.

9.3.3 File extension

The project owner wanted a file extension for files that are saved from the application. The functionality that a file extension '.mmv' is added to a file before saving was implemented in the application. Files with this extension are filtered out and are the only ones that are shown as default when saving and loading files in the Mmv application.

9.4 The Third Release

The result of iteration three is given below as figures. Figure 9.1 shows a text from an actual analysis window. Due to the fact that scaling and rotating the actual image of the window would have made it unreadable, it was presented this way instead.

As can be seen in figure 9.1, the three confidence intervals (95%, 97% and 99%) are presented beneath the R^2-value, which is presented beneath the results from the regression analysis. This is an example of a good regression analysis. The R^2-value is very near 1.0 which is an indication that the regression is well approximated to the given game data. Hence, the regressed third order polynomial that is presented is a good approximation of

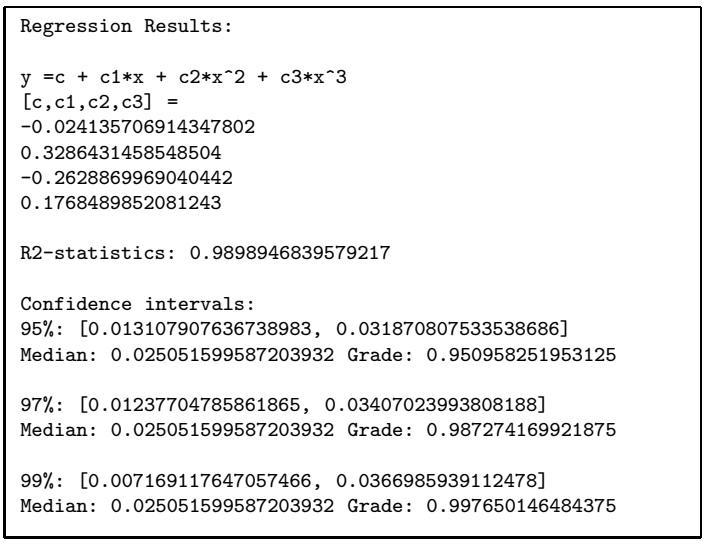

Figure 9.1: The analysis window with regression results.

the players utility function. The utility function that follows from the regression analysis is shown in figure 9.2.

The new game window is shown in figure 9.3. As can be seen, it has a check box to choose if the average mean value would be visible or not. There is also fewer buttons because the 'Ok' and 'Next' buttons are merged into one button with a double functionality. The 'Clear' button has been moved for a more clean and functional game window. It can also be seen that the probability fields has become wider, from three to four characters, to show more decimals.

9.4.1 Comments

When this release of the Mmv application was presented, the project owner was satisfied with it. This release had the right graphical appearance and functionality according to what the project owner wanted for the third release.

As this release is the last release of the project, it is also the final version of the application.

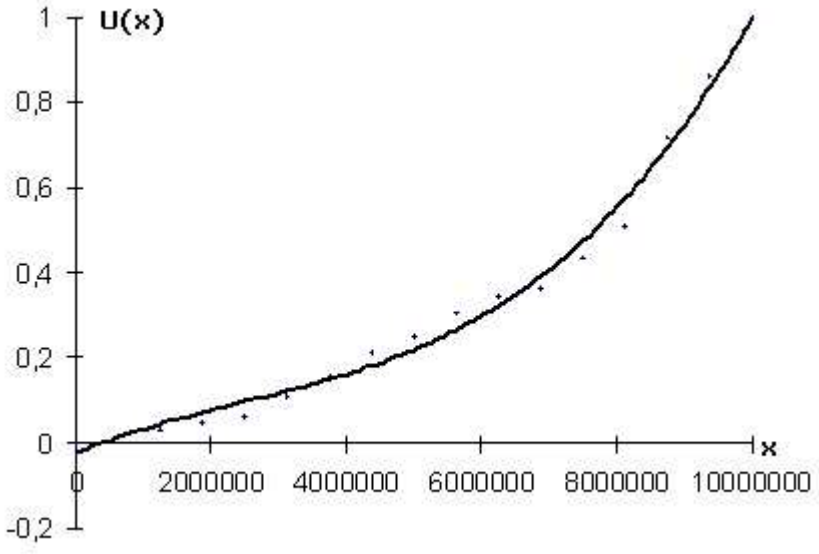

Figure 9.2: The utility function

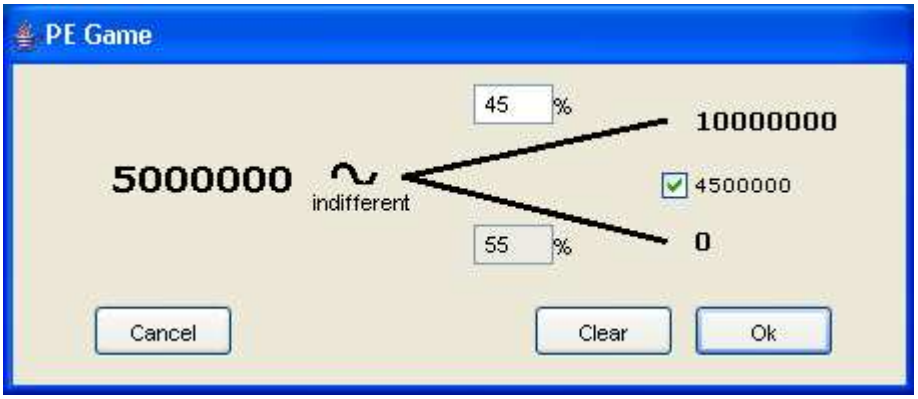

Figure 9.3: The game window for the third release.

Chapter 10

Thesis Summary

"The future is here. It's just not widely distributed yet."
– William Gibson (1948-)

10.1 Introduction

This chapter begins with a summary of the created application. The thesis is concluded followed by a discussion covering some of the problems and progress made during the project. Finally ideas of future works spawn from this thesis is presented.

10.2 MMV Summary

As described in section 8.3.3, the games in the Mmv application are created based on each other. All games, except the first, get their outcome values from a game already created or from the minimum and maximum outcome values decided for the game suite. The first game always get its outcome values from the minimum and maximum values decided for

the game suite, as demonstrated in figure 7.1.

This way of generating games raised some issues. As described later, inconsistent play is not possible, the first game played set the range for the following games. Also, it turned out that adding more games, did not generate a better utility function.

10.2.1 Additional Games Does Not Improve The Result

The game creation procedure and dependencies between the games resulted in a few aspects, like that an inconsequent result is impossible, and that the first three games played virtually sets the range for all other games.

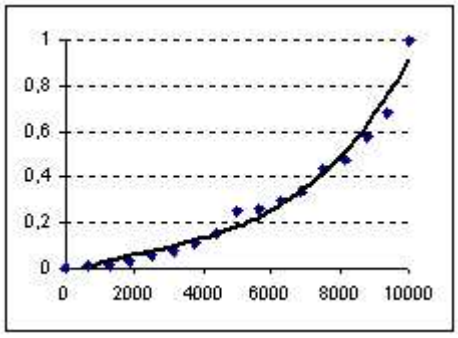

Figure 10.1: A game suite with 15 games played.

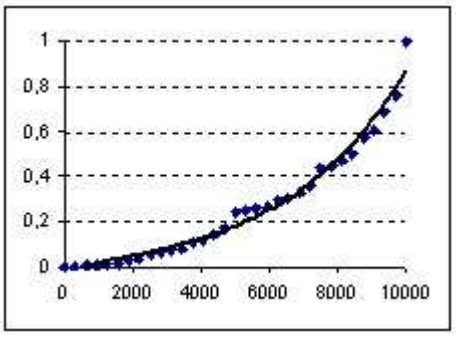

Figure 10.2: A game suite with 31 games played

In figures 10.1 and 10.2 two played game suites are shown. After playing 15 games the graph in figure 10.1 is the result from the regression. As can be seen in figure 10.2, playing another 16 games does not show any or only a small difference in the result of the regression. This is independent of the way the last 16 games are played. This is a consequence caused by the way the first games guide the utility.

10.2.2 The First Games Guide Utility

Since the utility calculations for the first game is based on the utilities 1.0 and 0.0 for the maximum and minimum outcome values respectively, the utility for the first game can be

any value in that interval. This implies that the probability set by the user for the first game, in a very high degree determines the resulting utility function. Let u be the utility of the first game, then the interval of the utility for the second and third game is $[0.0, u]$ and $[u, 1.0]$ respectively. Every game, partitions that interval for its respective sub game, in a similar way. All according to the theories in section 2.4.

After a while the interval becomes so small that all new sub games would obtain the same utility value as their parents.

10.2.3 Inconsistency Is Impossible

No matter how weird or bad a user plays the games, it is impossible for the user to be inconsistent. The way games are created forces the user to be consistent. Although the result may be a strange utility function, it is always consistent. The forced consistency comes from, as mentioned in section 2.4 that the player has to be consistent, that is, assign higher utility values to higher monetary values.

10.2.4 Solution

Would the games have been created in another way, the issues mentioned above would not have emerged. The games in each game suite can be better distributed if created independently of each other. As long as it is possible to calculate the utilities for all games, any game structure is possible.

Another method of creating games, could result in games where the player would be able to play inconsistently. If the player was allowed to play inconsistently, the inconsistencies must be resolved. French [7] says that it is up to the decision maker to resolve the inconsistencies. That would mean that checks would have to be implemented in the Mmv application to ensure that all inconsistencies are resolved. Perhaps by replaying the inconsistent games, or letting the user decide which games that are valid.

10.3 Contribution

The thesis contributes with a developed application, which can serve as a start for further development in the field of risk analysis. The theories described in the thesis have been implemented, and results in a risk attitude function.

Since some problems occurred with the use of a third order polynomial for representing the utility function, it might not be the most suitable in all occasions. Works derived from this one might need to investigate additional methods of representation.

This thesis instrumentalises the research article Zuccato[16] by describing the determination of the utility function, or preference function from that article.

As an additional contribution the thesis also shows that XP can be usable in a small development team, although some of the practices of XP are not usable in a two member team.

10.4 Experiences

This section discusses some experiences made throughout the project.

10.4.1 XP Experiences

To avoid unnecessary changes due to assumptions taken by developers, it seems vital in an XP based project to have the customer available when needed, or at least available within a short times notice. Perhaps a more detailed specification is needed if the customer is not available for a longer period. This is based on the fact that the customer changed the specification of the user interface design twice, once after each release.

These changes would probably not have occurred in an XP project with an on site customer to correct the assumptions of the developers at an early stage.

Since documentation of an XP project is done through code and not in massive paper

work, XP was suitable in a small team consisting of two members. However, automated documentation such as JavaDoc is good even in XP.

10.4.2 Regression Analysis

Since the R^2 value calculated after the analysis is done usually is close to 1, suggests that a correct model has been used. How well the approximation of the third order polynomial is done depends on how good the game is actually played. For instance, if an user sets the probability to 0% or 100% in all games, then the utility values will not generate a good approximation, because the utility values do not follow a third order polynomial, in this case the data should have been approximated to another model.

10.5 Conclusion

To conclude this thesis, a look at the objectives presented in section 1.1 is in order.

The first objective of the thesis was to create a Java-application to determine a person's risk attitude, and that the application would be developed using the software development method of XP. The first objective is met by the implementation of the Mmv application described in chapters 6 through 9. This application has been developed using the XP practises mentioned in section 4.2.

The second objective required the development of the application to be described, as well as the theories used to determine risk attitudes. This objective is met by this document, that describes both the creation of the Mmv application and the theories of risk analysis, risk attitudes and utility functions as well as regression analysis.

10.6 Future Work

Further development of the application developed in this thesis is possible. For instance modifying the game structure to avoid the issues mentioned in section 10.2.

Based on the research article, Zuccato [16], another application can be developed where different risk scenarios can be assessed. When these scenarios have been assessed, the results from the Mmv application can be used by a person other than the decision maker that played the games, using the modified mean value approach. This person can then act like as if the decision maker had been faced with the scenarios and had taken the relevant actions.

References

[1] Robert A Adams. *Calculus - A Complete Course*. Addison Wesley, 5th edition, 2003.

[2] Howard Anton. *Elementary Linear Algebra*. John Wiley & Sons, Inc, 8th edition, 2000.

[3] Kent Beck and Cyntia Andres. *Extreme Programming Explained*. Addison Wesley, 2nd edition, 2005.

[4] Robert V. Binder. *Testing Object-Oriented Software*. Addison Wesley, 1999.

[5] Han Bleichrodt, Jose Luis Pinto, and Peter P. Wakker. Making descriptive use of prospect theory to improve the prescriptive use of expected utility. *Management Science*, 47(11):1498–1514, 2001.

[6] Norma R. Draper and Harry Smith. *Applied Regression Analysis*. John Wiley & Sons, Inc, 3rd edition, 1998.

[7] Simon French. *Decision Theory*. Ellis Horwood Limited, 1st edition, 1993.

[8] Steve Frosdick. The techniques of risk analysis are insufficent in themselves. *Disaster Prevention and Management*, 6(3):165–177, 1997.

[9] Erich Gamma, Ralf Johnson, Richard Helm, and John Vlissides. *Design Patterns*. Addison Wesley, 2004.

[10] Gilbert Gordon and Israel Pressman. *Quantitative Decision-Making For Business*. Prentice Hall, 2nd edition, 1983.

[11] Ian Sommerville. *Software Engineering*. Addison Wesley, 5th edition, 1995.

[12] Doug Tidwell. *XSLT*. O'Reilly, 1st edition, 2001.

[13] Kerstin Vännmann. *Matematisk statistik*. Studentlitteratur, 2nd edition, 2002.

[14] The World Wide Web Consortium W3C. The extensible markup language (xml) 1.0 3rd edition. Available on the Internet:http://www.w3.org/TR/2004/REC-xml-20040204/, 2004.

[15] Mark Allen Weiss. *Data Structures & Algorithm Analysis in C++*. Addison Wesley, 2nd edition, 1999.

[16] Albin Zuccato. A modified mean value approach to assess security risks. In L. Labuschagne and M. Eloff, editors, *2nd Annual Conference of Information Security for South Africa – ISSA-2 Proceedings*. South African Computer Society, 2002.

Appendix A

XP Stories

A.1 Prototype 1

This part lists the XP-stories for the first prototype.

Story: Story1 Statusbar **Nr:** 1 **Priority:** 4

Description: The statusbar, should show last action for 15 seconds.

Story: Story2 Close program **Nr:** 2 **Priority:** 3

Description: Closing program should ask to save.

Story: Story3 About **Nr:** 3 **Priority:** 2

Description: About should contain version and copyright notice.

Story: Story3 Open File **Nr:** 4 **Priority:** 3

Description: Filter for application file type, and memorize last path.

Story: Story3 Menus **Nr:** 5 **Priority:** 1

Description: The structure of the menus should look as in table A.1.

File	New Session Load Session Dave Session Import Export Exit
Edit	
Display	Graph Table
Data Handling	
Window	
Help	Help About

Table A.1: Menu structure

A.2 Prototype 2

Story: Input **Nr:** 6 **Priority:** 1

Description: Input values for regression analysis.

Task: Values **Story:** 6 **Priority:** 1 **Time:** 4 hours

Description: The user should be able to enter a set of values.

Story: Regression **Nr:** 7 **Priority:** 1

Description: Regress some values.

Task: Regression **Story:** 7 **Priority:** 1 **Time:** 10 hours

Description: Perform a regression analysis, to fit a third order polynomial to the inputed

values from the task Values.

A.3 MMV

Story: Game **Nr:** 8 **Priority:** 1

Description: 20 games should be playable.

Task: Lottery **Story:** 8 **Priority:** 1 **Time:** 16 hours

Description: 20 PE games should be playable.

Story: Menu structure **Nr:** 9 **Priority:** 1

Description: Menu structure without functionality.

Task: Create Menu structure **Story:** 9 **Priority:** 1 **Time:** 2 hours

Description: The menu structure was supposed to be similar to that in the user interface prototype.

Story: User Interface **Nr:** 10 **Priority:** 1

Description: The user interface should be similar to that of the user interface prototype.

Task: Statusbar **Story:** 10 **Priority:** 1 **Time:** 2 hours

Description: The application should have the same statusbar as the user interface prototype. That is it should show its content for 15 seconds and then clear itself.

Task: Close Program **Story:** 10 **Priority:** 1 **Time:** 2 hours

Description: When a user tries to close the program, a dialog should ask whether to quit or save any unsaved data before quitting.

Task: About **Story:** 10 **Priority:** 1 **Time:** 2 hours

Description: The text for the about dialog should come from a file.

Story: Regression **Nr:** 11 **Priority:** 1

Description: A third order polynomial should be fitted to gamedata.

Story: Statistical Tests **Nr:** 12 **Priority:** 3

Description: Various tests should be performed on the result from the regression, to determine if it represents a sound model according to gamedata.

Story: Save Game Result **Nr:** 12 **Priority:** 2

Description: After the user has successfully played games, or the save menu option is chosen, the results of the games should be saved. The file saved should be in a XML format.

Task: Design a DTD **Story:** 12 **Priority:** 2 **Time:** 4 hours

Description: The DDT for the file to save must be designed, to allow games to be saved.

Task: Saving Games **Story:** 12 **Priority:** 2 **Time:** 8 hours

Description: Save games and the fitted utility function for the games.

Story: Game Settings **Nr:** 13 **Priority:** 2

Description: To get some versity in the games, various gameparameters should be setable before starting a new game. These settings should be read and updated from a parameter file.

Task: Design a parameter file DTD **Story:** 13 **Priority:** 2 **Time:** 1 hours
Description: The DDT for the parameter file.

Task: Reading the parameter file **Story:** 13 **Priority:** 2 **Time:** 4 hours
Description: Read the parameter file.

Task: Create a user interface **Story:** 13 **Priority:** 2 **Time:** 4 hours
Description: The user interface should allow to set various settings for the game.

Task: Update the parameter file **Story:** 13 **Priority:** 2 **Time:** 2 hours
Description: Update the parameter file through the user interface.

Story: Import and Export **Nr:** 14 **Priority:** 3

Description: It should be possible to import and export data to and from the application. The import and export system should allow for easy extendability. Therefore the Abstract factory, or an equivalent design is needed.

Task: Design of Abstract Factory **Story:** 14 **Priority:** 2 **Time:** 4 hours
Description: Design a plugin based framework using the Abstract factory.

Task: XML importer **Story:** 14 **Priority:** 3 **Time:** 4 hours
Description: Import data using XML

Task: XML exporter **Story:** 14 **Priority:** 3 **Time:** 4 hours
Description: Export data using XML

Story: Statistical Tests **Nr:** 15 **Priority:** 3

Description: To check whether the result from regression is well fitted, some statistical tests should be done on the result.

Task: R^2-test **Story:** 15 **Priority:** 2 **Time:** 8 hours

Description: Test the regression using a R^2-test.

Task: Confidence Interval **Story:** 15 **Priority:** 2 **Time:** 8 hours

Description: Test the residuals using statistical, confidence interval

A.4 Additional stories

Story: Changes Release 2 **Nr:** 16 **Priority:** 1

Description: Change code according to comments from release 1

Task: Adding symbol '%' **Story:** 16 **Priority:** 1 **Time:** .1 hours

Description: The symbol '%' should be put to the right of the two probability fields.

Task: Add average mean value **Story:** 16 **Priority:** 1 **Time:** .1 hours

Description: Add an average mean value between the two outcomes.

Task: Add buttons **Story:** 16 **Priority:** 1 **Time:** .3 hours

Description: Add a button with the same functionality as the 'Enter' key in the probability field. This button should have the caption 'Ok'. Add a button for clearing the probability field. This button should have the caption 'Clear'. Change location of the

buttons for easier usage.

Task: Change caption of 'Stop' **Story:** 16 **Priority:** 1 **Time:** .1 hours
Description: The caption of the 'Stop'-button should be changed to 'Cancel'.

Task: Add indifference symbol **Story:** 16 **Priority:** 1 **Time:** .2 hours
Description: Add a symbol for indifference above the 'indifferent' text.

Task: Two different values in fields **Story:** 16 **Priority:** 1 **Time:** .5 hours
Description: Add the possibility to enter the probability value in two different manners, as a decimal between 0 and 1, or as an integer between 0 and 100. The error message that is shown when a user enters an illegal value should also be changed due to this fact.

Task: Raise outcome values **Story:** 16 **Priority:** 1 **Time:** .1 hours
Description: Give the oucomes higher values.

Task: Game distribution **Story:** 16 **Priority:** 1 **Time:** .5 hours
Description: The games should be more randomly distributed.

Story: Changes Release 3 **Nr:** 17 **Priority:** 1

Description: Change code according to comments from release 2

Task: Probability fields **Story:** 17 **Priority:** 1 **Time:** .1 hours
Description: Probability fields should have 4 characters instead of 3.

Task: Checkbox for average mean value **Story:** 17 **Priority:** 1 **Time:** .2 hours
Description: Add a checkbox for average mean value between the two outcomes.

Task: File extension **Story:** 17 **Priority:** 1 **Time:** .2 hours

Description: The file extension '.mmv' should be added to files that are saved from the application.

Task: Filter file extension when loading **Story:** 17 **Priority:** 1 **Time:** .2 hours

Description: The file extension '.mmv' should be filtered out when loading files.

Task: Ok + Next **Story:** 17 **Priority:** 1 **Time:** .5 hours

Description: The two buttons 'Ok' and 'Next' should be changed to one button with a double functionality. When there is something entered in the probability field then the button should have the same functionality as the former 'Ok' button. After the user has confirmed the inputted value, either by pressing the enter key or using the 'Ok' button, the button should have the same functionality as the former 'Next' button. When pressing the button in the latest mode the user can play the next game.

Appendix B

UML-Diagrams

The UML-diagrams shown here shows the class-diagrams for each Java package, as well as the package structure. There is also a diagram showing the dependencies between all classes in the application. The diagrams were generated automatically using reverse engineering in Borland Together 6.1. As the diagrams were generated from code, they meet the XP need, of only having automatically generated documentation to ensure that the documentation always reflect the latest source.

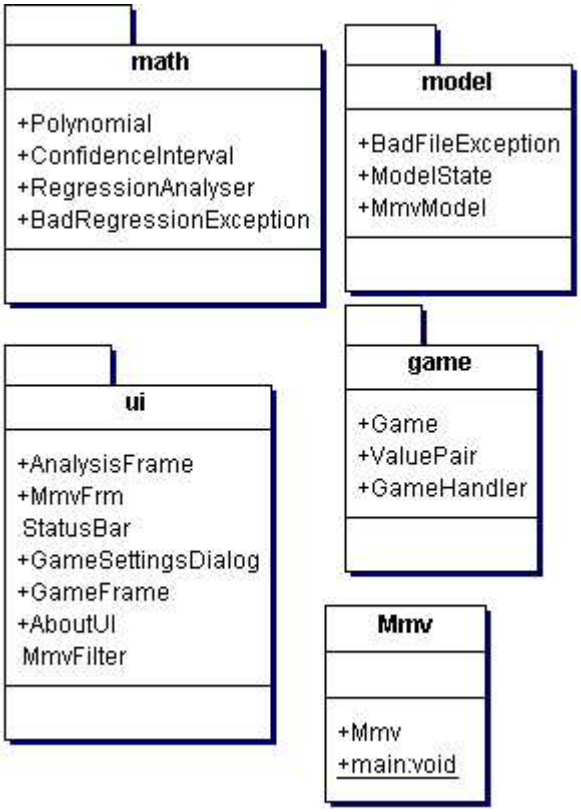

Figure B.1: Overview of all packages.

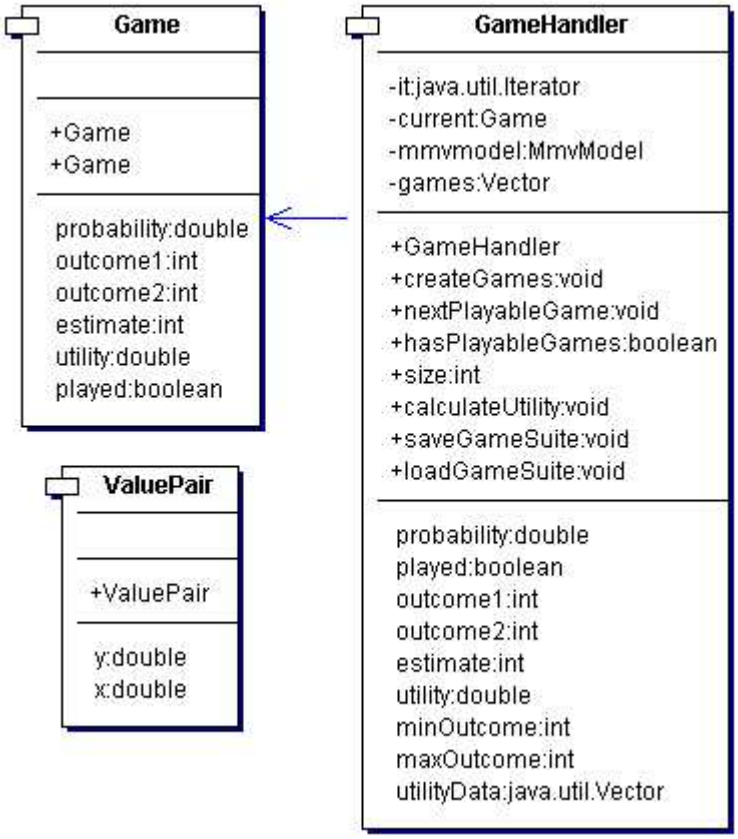

Figure B.2: The parts of the game package.

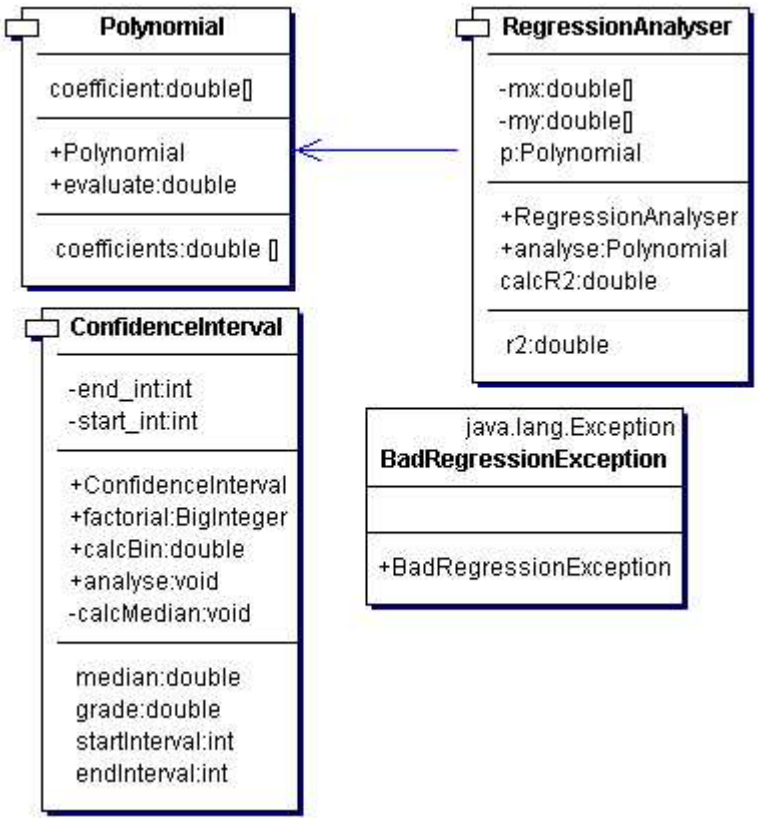

Figure B.3: The parts of the math package.

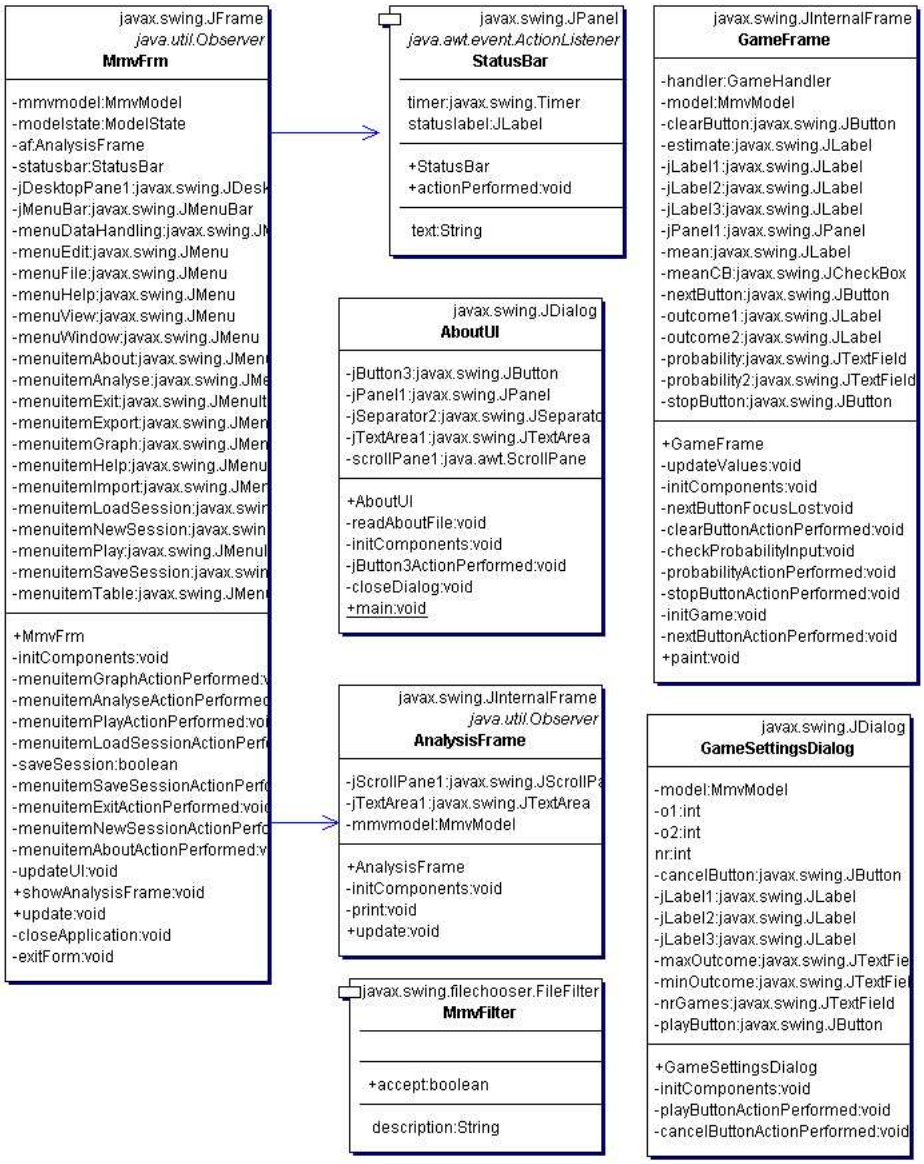

Figure B.4: The parts of the ui package.

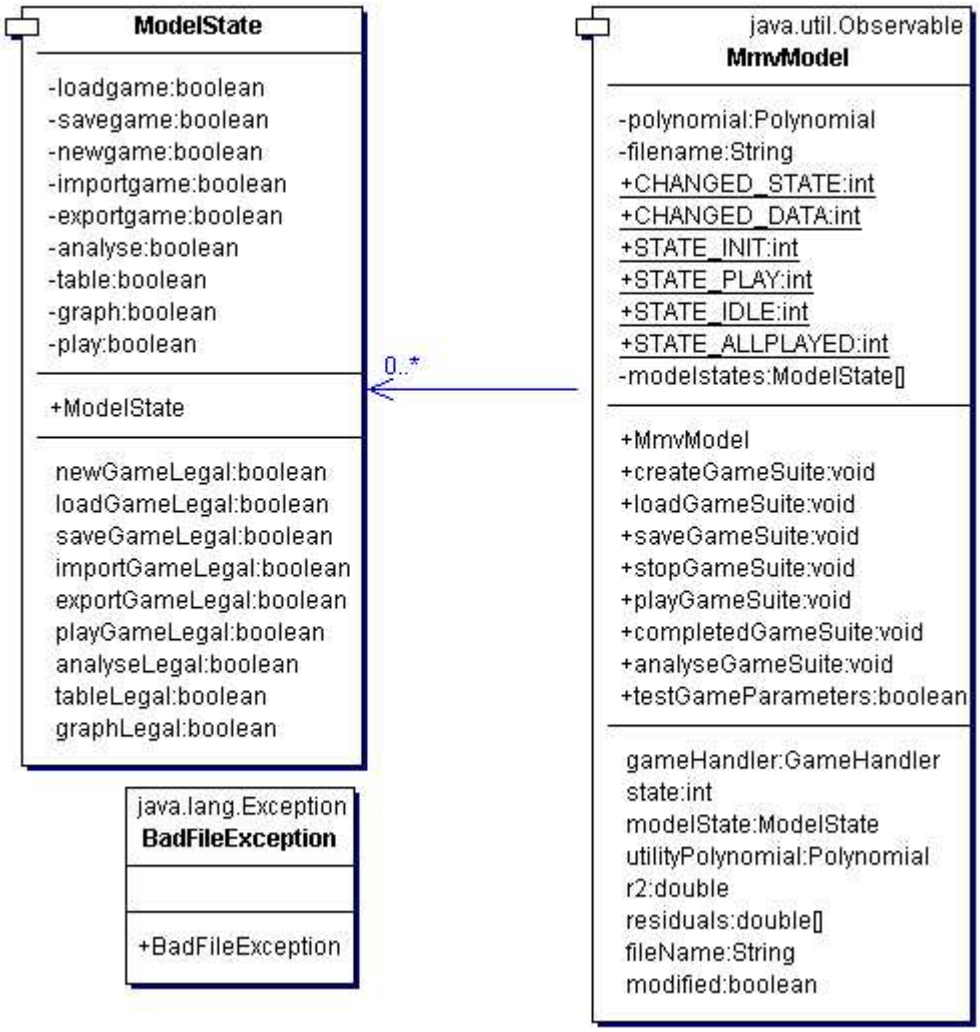

Figure B.5: The parts of the model package.

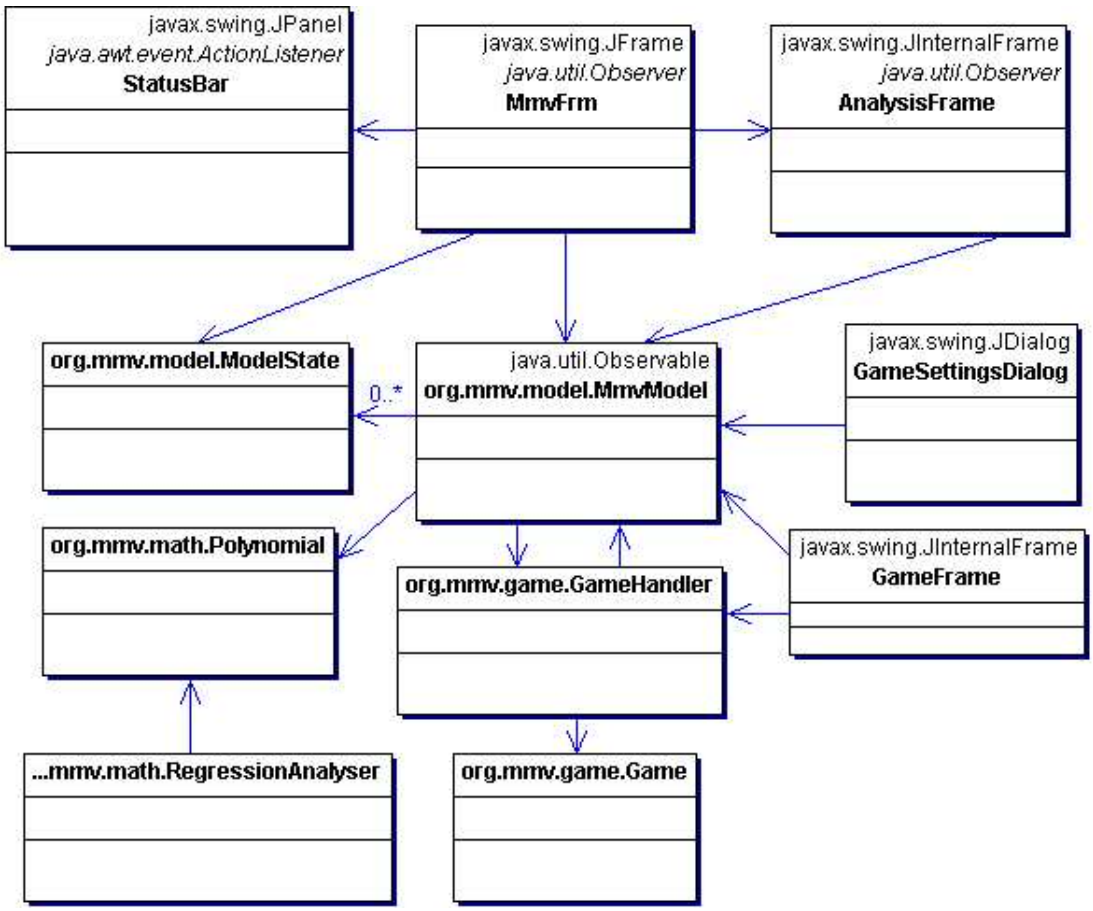

Figure B.6: Overview of all classes and their relationship.