Charles University in Prague

Faculty of Social Sciences
Institute of Economic Studies

BACHELOR THESIS

Agent-Based Model of the Software Market

Author: **Michal Bureš**

Supervisor: **PhDr. Jiří Kukačka**

Academic Year: **2014/2015**

Declaration of Authorship

The author hereby declares that he compiled this thesis independently, using only the listed resources and literature.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis document in whole or in part.

Prague, May 13, 2015

Signature

Acknowledgments

I am indebted to PhDr. Jiří Kukačka, the supervisor of this thesis, for introducing me into the field of agent-based modelling and for his valuable help. I am also grateful to my parents without whom my studies would not be possible.

Abstract

In this thesis we apply agent-based modelling methodology on the software market. We derive and implement an original model of the market as the research in this field seems negligible. The work focuses mainly on the derivation of the model and explanation of the methodology. Specific features of the software market are discussed and presented as assumptions of the model. Based on these principles we construct a model of the software market with complex customer behaviour. In the end we test our model in an application we developed solely for this purpose. Responses of the model to dynamic modifications of individual parameters are tested using the application. A simple case scenario in which we apply the model on the dynamic market of game engines is presented.

JEL Classification	C63, D11, D47, D82, D83, L10
Keywords	Software market, Simulation, Agent-based modelling, Customer behaviour

Author's e-mail	`bures.michal@hotmail.com`
Supervisor's e-mail	`jiri.kukacka@gmail.com`

Abstrakt

Tato bakalářská práce je aplikací agentního modelování na trh softwarových produktů. Našim cílem je odvodit a implementovat nový model softwarového trhu, jelikož výzkum v této oblasti je zanedbatelný. Text práce se zabývá především odvozením modelu a popisem metodologie. Také jsou diskutovány specifické rysy trhu softwarových produktů, které jsou použity jako předpoklady modelu. Na jejich základu navrhneme model softwarového trhu s komplexním chováním zákazníků. Model je testován pomocí aplikace implementované za tímto účelem. Pomocí této aplikace jsou testovány reakce modelu na dynamické změny parametrů. Součástí práce je také scénář, kde aplikujeme náš model na trh herních enginů.

Klasifikace JEL	C63, D11, D47, D82, D83, L10
Klíčová slova	Trh softwarových produktů, Simulace, Agentní modelování, Chování zákazníka

E-mail autora	`bures.michal@hotmail.com`
E-mail vedoucího práce	`jiri.kukacka@gmail.com`

Contents

List of Tables viii

List of Figures ix

Acronyms x

Thesis Proposal xi

1 Introduction 1

2 Software Market 3

2.1 How Does a Customer Decide? 6

3 Agent-Based Modelling 7

3.1 Definition of ABM 7

3.2 Variety of Applications 8

3.3 ABM in Economics and Finance 9

3.4 Spread and Justification of ABM 11

3.5 Implementation of ABM and OOP 12

3.6 Validation of ABMs 14

4 Model of the Software Market 15

4.1 Structure 16

4.2 Model Advantages 17

4.3 Customer Agents 17

4.3.1 Initialization 18

4.3.2 Parameters 18

4.4 Agent Behaviour 18

4.4.1 Motivation 19

4.4.2 Perception 20

4.4.3 Satisfaction 21
4.4.4 Willingness to Change 22
4.4.5 Decision 23
4.5 Product 24
4.6 The Big Picture 26

5 Implementation and Model Analysis 27
5.1 Model Application Interface 27
5.2 Calibrating the Model 30
5.3 Observing the Effects 31
5.3.1 Price Effect 32
5.3.2 Marketing Effect 33
5.3.3 Quality Effect 34
5.3.4 Combined Effect 35
5.3.5 Price Wars 36

6 Case Scenario - Game Engines 39

7 Conclusion 43

Bibliography 47

A Important Code I

B Content of Enclosed Zip Archive IV

List of Tables

3.1 OOP and ABM Connection 13

4.1 Parameters of the Customer Group 19
4.2 Product Parameters 26
4.3 Simulation Actions 26
4.4 Customer Actions 26

5.1 Reasonable Values of Customer Parameters 30
5.2 Reasonable Values of Product Parameters 31
5.3 Settings of the Test Scenario - Customer Groups 31
5.4 Settings of the Test Scenario - Products 32

6.1 Settings of the Game Engines Scenario - Customer Groups . . . 39
6.2 Settings of the Game Engines Scenario - Products 40

List of Figures

3.1 Example of an Agent in OOP 13

4.1 Theoretical Basis of the Model 16
4.2 Simulation Settings Dialog 18
4.3 Graph of $arctan(x)$ 23
4.4 Graph of $f(S_{i,t})$ 24
4.5 Simulation Diagram 25

5.1 Settings Window 28
5.2 Control Window 28
5.3 Output Window 29
5.4 Sample of Exported CSV Product Data 30
5.5 Price Effect - Decrease 32
5.6 Price Effect - Increase 33
5.7 Marketing Effect - Decrease 33
5.8 Marketing Effect - Start Decrease 34
5.9 Quality Effect - Decrease 34
5.10 Quality Effect - Increase 35
5.11 Combined Effect - Marketing Decrease, Quality Increase 36
5.12 Price Wars 36
5.13 Price Wars with Additional Steps 37
5.14 Price Wars 2 37

6.1 Game Engines Scenario - No Modifications 40
6.2 Game Engines Scenario - Dynamic Modifications 41
6.3 Game Engines Scenario - Dynamic Modifications 2 41

Acronyms

2D	Two Dimensional
3D	Three Dimensional
.NET	Software framework from Microsoft
ACE	Agent-Based Computational Economics
ABM	Agent-Based Modelling/Model
CSV	Comma Separated Values
DSGE	Dynamic Stochastic General Equilibrium
EURACE	Macroeconomic Model of European Economy
HAM	Heterogenous Agent-Based Model
NetLogo	Multi-Agent Programmable Modelling Environment
OOP	Object Oriented Programming
WPF	Windows Presentation Foundation

Bachelor Thesis Proposal

Author	Michal Bureš
Supervisor	PhDr. Jiří Kukačka
Proposed topic	Agent-Based Model of the Software Market

Topic characteristics

This thesis should be an application of agent-based modelling methodology to the study of the market for software products. The products on the market will serve the same purpose, but qualitative parameters will differ. The price of the products can also vary as it is the key in determining how the market forces interact.

In the model few firms compete for customers to maximize their profit, or even to achieve more complicated objectives. The product has preset characteristics making it unique among others as is common in reality. Each firm is free to dynamically change the price of their product. The simulation will proceed in discrete steps. In each step firms make their decisions and present their offers to the market. The agents make their choice based on their preferences including the product's price. They will also remember their own user experience making those, who are unsatisfied, more likely to change their choice. The central interest of the simulation will lie in the choice of the pricing strategy of an individual firm. A special attention will also be given to the market share and profit dynamics. In the second part I will focus on the application of the model. Interesting hypotheses connected with pricing strategies will be tested by employing this unique model.

Hypotheses

1. Agent-based modelling is a suitable methodology for software pricing and customers modelling.
2. Optimal pricing strategies differ significantly among heterogenous products.

3. Freemium model can be more profitable than conventional pricing strategies.
4. If we assume profit as a function of price, it is common that the function has multiple local maxima.
5. It is reasonable to change the price of the product during its life cycle.
6. Software as a service can be the optimal strategy for products with certain characteristics.

Methodology

A simulation model based on agent-based modelling best practices will be developed to represent the market for software products. The structure of the model will be developed specifically to realistically represent the market for software products. C#, a modern object-oriented programming language, will be used to write a stand-alone simulation.

Outline

1 Pricing Strategies
1.1 Motivation
1.2 Product pricing strategies
1.3 Software pricing
2 Agent-Based Modelling
2.1 ABM methodology
2.2 ABM and pricing strategies
3 The Model
3.1 Theoretical Explanation
3.2 Implementation
4 Evaluating Strategies
4.1 Using the model
4.2 Evaluating strategies
4.3 Interpretation
4.4 Results
5 Conclusion

Core bibliography

1. Alkemade, F. (2004): "Evolutionary Agent-Based Economics." *Technische Universiteit Eindhoven.*

2. Lehmann, S. & P. Buxmann (2009)): "Pricing strategies of software vendors." *Business & Information Systems Engineering* **1(6)**: pp. 452-462.

3. Macy, M. W. & R. Willer (2002): "From factors to actors: Computational sociology and agent-based modeling." *Annual Review of Sociology* **28**: pp. 143-166.

4. Rao, V. (2010): "Handbook of Pricing Research in Marketing." *Edward Elgar Publishing.*

5. Schildt, H. (2008): "C# 3.0 The Complete Reference 3/E,." *McGraw-Hill Education.*

6. Tesfatsion, L. (2006): "Agent-Based Computational Economics: A Constructive Approach to Economic Theory." *Staff General Research Papers 12514*, Iowa State University, Department of Economics.

7. Tesfatsion, L. & K. L. Judd (2006): "Handbook of Computational Economics." *Vol. 2 of Handbook of Computational Economics*, Elsevier.

8. Ondřej Šerý, Tomáš Poch, Pavel Šafrata & Cyril Brom (2006): "Level-of-Detail in Behaviour of Virtual Humans." *Volume 3831 of Lectures in Computer Science*, Springer Berlin Heidelberg.

Chapter 1

Introduction

The state of technology is moving forward in a pace unseen by the past. New, structurally different markets are emerging and innovation sparks various forms of interaction. Existing markets are not excluded from this development. The rapid growth of information processing, data analysis and automation induces new, more effective means of matching supply and demand. Transaction costs are lowering and sellers can be more direct in their access to customers. It is a great challenge for economists to capture and explore the mechanics of both the brand new and the changing markets. Software market is one of these new markets. Despite the fact, that it is one of the most innovative markets, it lies on the margin of research attention.

The objective of this thesis is to explore the mechanics of the software market by developing an agent-based model that captures the processes of the market for software products with a common purpose. It is constructed in accord with the ABM methodology. The agents in the model are built based on the basics of microeconomic theory. Simulational approach offers us a possibility mostly unavailable in theoretical models. The agents can be more than a purely abstract entity with a static equation behind. They can learn, remember the degree of their satisfaction and the results of their decisions. This form of dynamic individuality should be the key concept and advantage of this model. The structure of our model is inspired by the model in Zhang & Zhang (2007). Antonides (1989) and Said *et al.* (2001) serve as a reference for my reasoning about the decision making system of customer agents.

The thesis is structured as follows. At first, the basic characteristics of the software market are explored in Chapter 2. Together with an inquiry into customer behaviour, they serve as a basis for deriving a simplified representation

of the market. Agent-based modelling methodology is reviewed in Chapter 3 where examples of existing models are presented. Chapter 4 is the core of this thesis. It is concerned by the theoretical derivation of the model of the software market. A standalone simulation is written to enable usage of the model. In Chapter 5 the implementation of the simulation and its usage are described. Individual effects of dynamic parameter modifications are tested. Chapter 6 follows with a small case scenario based on the dynamic market for game engines. Finally, the contribution of our work is summarized.

Chapter 2

Software Market

In this chapter we study the software market and formulate its characteristics. The software market is a market where software products are sold. We are interested in the form of production and distribution of software and its key features. Nowadays the distribution of software is done mostly via the Internet. Anyone can run free software often with less than ten mouse clicks. Payments are done via the Internet within seconds. Software is nothing more physically than a sequence of zeros and ones. But multiple levels of abstraction enable programmers to write complex programs that bring enormous increase in effectivity of repetitive labour. Contemporary programs can often do the job of a few people, or even a job that could not be done by any human. The precision and speed are the domains of the computer however creativity still lies in the hands of men. The greatest advantage of software is something that has been known for hundreds of years. It is connected with the invention of the printing press around 1440. Books were written at first with a large amount of effort. But thanks to the spread of printing press the book could reach a huge amount of readers with a reduced price. Then you can imagine a car designer creating a concept for a new car. The process is costly and it is the core creative process in development of a new car. Let us assume the company spends 500 000 $ for a concept of a car. It is a lot of money but production of every car may cost few thousands of dollars. This reveals the change in the characteristics of effectivity. When Smith (1776) was written it was mainly the effectivity of production that was important. Smith spent many pages describing the importance of the division of labour. With printing books and now even more with selling software we face the phenomenon of duplication. From this we derive two important assumptions for our model.

Negligible transaction costs. Most of the software products are available online either for direct use or with easy download. Customers are free to gather information on the internet and face little or no barriers when they want to acquire a specific product. The process of buying often comes down to paying for a license. Product is then ready to use. It is straight-forward and fast.

The software market has **low variable costs**. Once a software product is developed a customer support needs to be provided. But this depends only weakly on the amount of copies sold. Because the variable costs are significantly lower than fixed costs the best strategy is maximizing income.

Continuing with our analysis of the market we capture another important characteristic. On the software market the price of an individual product is not very high. The choice of the customer when buying a new car depends dramatically on his income and wealth. When buying a software product the customer is influenced by those characteristics only slightly. That means we can expect **linear price elasticity** on the software market. Additional unit paid for the product brings about the same loss to our customer as the individual prices tend to be low. This is supported by the fact, that he buys only one instance of the product. So we do not have to take into account the diminishing utility of buying another unit.

On the software market we assume **highly differentiated products**. Every product has its name, brand and product characteristics. Generally, only one instance of the product is bought and the customer views it as different. The structure of the market is specific. In fact it consists of a huge amount of small software markets. They influence each other in certain ways. Events such as adoption of one product can increase the need of another product, or just improve the perception of a related product. But mostly the products are independent and their qualities are not very correlated. So we assume that the structure of the software market resembles a large amount of **independent, but similarly functioning markets**. We can view them as separated without a large loss of realism.

We assume that the individual markets for software products have **oligopolistic market structure**. Developing a qualitatively capable software product is a complex and difficult task. It requires work of many programmers and testers to deliver a successful product. Managers and sales analysts are needed to maximize the profit from an existing product. There is a plenty of opportunities for developing a software product. So when the market for a product with a specific purpose is satiated, it is likely, that the company will choose another

market to penetrate. So there exist only a few software products for every specific purpose as it is not profitable trying to penetrate satiated markets.

When customer chooses a product many important aspects play a role in his decision. Price is an important parameter of his decision as on other markets. When buying bananas the customer often remembers his past experience. If he bought delicious bananas last week in a shop nearby he will repeat his choice with a greater probability. On the contrary, if he saw rotten bananas in the shop nearby he is likely to avoid the shop, at least for some amount of time. With software this effect is even stronger. Any experience the customer has with a specific software product is the most important information he has about the product. Using a software product happily for a year, the customer will probably continue to use it unless the underlying situation changes. He can perceive a new need, or a new version of a concurring product with a lot of new functionality can reach the market. Often we see emergence of products that are free, at least in some degree. Even after such an impulse, the customer may stick with the product he uses simply because he is satisfied and is not willing to spend time making a new choice. So we observe an **important role of experience and memory.**

Software products are very complicated and their features and quality can take a long time to recognize. The customer often realizes that the product lacks some qualities only after weeks of using it. Discovery of serious flaws of the product can cause a strong incentive to look for another product that will better suit the needs of the customer. On the contrary, the influence of satisfaction of the product is a reason to stick with the product though we suppose that is has smaller impact than unsatisfactory product. **High influence of unsatisfying quality** is another assumption we make about the software market.

Another fact that is connected with the complexity of software product lies in the hardihood with which the customer reviews the product. That induces him to search for helpful information wherever he can. Aside from the information on the Internet, or in a shop the customer tries to gather information in his social network. He will ask friends about their opinion on the product. They can tell him about the experiences with the usage of their product and the customer will take this information into account when making his decision. So we can see that **influence of friends** has its place in the decisions of the customer.

2.1 How Does a Customer Decide?

On the software market the decision of the customer is often a long and complicated process. As we mentioned before the products are differentiated. When the customer makes his choice he has to make a comparison of the available products. We assume that the customer can assign a certain value representing his utility or preference for the product. This is an important assumption. Without it we would find it very complicated to simulate his behaviour. So existence of a function that assigns the utility value is required:

$$f_u(p_1, p_2, ..., p_n, c_1, c_2, ..., c_m) \tag{2.1}$$

where $p_1, p_2, ..., p_n$ are parameters of the product that influence the perceived utility. Similarly $c_2, ..., c_m$ are parameters of the customer on which the utility depends. In Chapter 4 we build on this requirement when describing our motivation function.

Chapter 3

Agent-Based Modelling

Agent-based modelling is a promising methodology that is trying to make its way to economic analysis. It is applied in a wide range of scientific areas. The rise of ABM is connected with increased computational capability of modern computers. It is only natural that many scientist try to utilize computers in their research. At first, there was no common methodology, but the struggle for effectivity enabled the emergence of shared knowledge. So a set of guidelines and theory concerning representation of reality came into being under the headline of agent-based modelling.

It is a theoretical background for computer simulations that tries to capture the behaviour of non-trivial agents. According to Epstein (1999) ABM models provide computational demonstrations that a given microspecification is in fact sufficient to generate a macrostructure of interest. Recently, in Macal & North (2010) ABM is described as a new approach to modelling complex systems composed of interacting and autonomous agents.

ABM is offered as a solution of the failure of mainstream economic models. Farmer & Foley (2009) react to the inability of DSGE models to predict crises and sudden shocks. Bouchaud (2008) by criticism of mainstream economics calls for a different paradigm in economic modelling. He gives hope to econophysics and behavioral economics.

3.1 Definition of ABM

Most definitions of agent-based models contain three central features as stated in Windrum *et al.* (2007):

1. Bottom-up approach: In agent-based models we want to derive macro

results from the micro behaviour of individual agents. This is often set in contrast with the so called top-down approach of the neoclassical models of mainstream economics. Clasically the bottom level is represented by an individual with constraints connected with reaching of equilibrium. In ABM we have heterogeneous agents that form complicated systems. They evolve in time based on complex interactions. The macro outcome is then determined by the many actions taken by the individual agents (Epstein & Axtell 1996).

2. Boundedly-rational agents: It is not possible to implement agents as fully rational entities. That would be beyond capabilities of any imagainable computational model. That means we have to substitute rationality with some other approach. We arrive at some partly rational rules, often in form of an optimisation problem. We can apply some heuristics based on observation from the real world. So in ABMs agents are modeled as boundedly rational forming a compromise between rationality and viability.

3. Network directed interactions: The interaction among the individual agents should be direct and non-linear. The interactions are direct because the particular decision depends straightly on the past choices made by other agents present in the simulated population. Groups of agents interacting among themselves or networks based on spatial proximity can serve as examples (Irwin & Geoghegan 2001). Structure of these interactions may change over time, as the agents develop in a dynamic way.

These features when combined can lead to complex outcomes. Through the aggregation process structurally new objects or actions can emerge. Agents inherently do not have any understanding of the environment. But through interactions in agent-based models a learning process can be implemented.

3.2 Variety of Applications

The variety of possibilites to use agent-based models is enormous. We choose only three representative interesting examples to ilustrate the breadth of applications.

In Castella *et al.* (2005) an interesting agent-based model of land-use in mountainous Vietnam is described. It is concerned mainly with scale factors and their effect on the model. The author uses a visual represantation and calibrates the model using old aerospatial photographs. The model is run with different spatial resolutions. The land-use preference is found to be largely affected by the scale.

Auchincloss *et al.* (2011) tries to explore the role of economic segregation in causing income disparities in a diet. By using a simple agent-based model a relationship between income differentials and the segregation of low-income households and unhealthy food stores from high-income households and healthy food stores is discovered. In the low income group, increasing their preferences for healthy food improved their diet but still a difference remained.

Epstein (2009) discusses the application of ABM on the modelling of the spread of the swine flu disease. It is a logical continuation of the classical disease spread modelling. Irrational behaviour and complex networks can be modelled on a global scale using ABM.

In Axelrod (1997) a list of simple replicable models can be found.

3.3 ABM in Economics and Finance

In recent years agent-based modelling capabilities are being used more widely in the field of economics. One of the most common applications of ABM is modelling financial markets. There from the ABM methodology heterogeneous agent models or simply HAMs are derived. In Amman *et al.* (2006) its use is explored. They argue that in economics we can see an important paradigm shift. Traditionally, in economics representative rational agents were used. They had access to full information and through some maximization problem they hace chosen the best suited option available. Here a new agent-based approach is advocated. The agents are only boundedly rational and make decision based on simple rules of thumb. These come mainly from the field of behavioural economics. They support this attitude with the view of Kahneman and Tversky who propose that the behaviour of agents under uncertainty can be best described by simple heuristics and biases. They support it by the role expectations and crowd behaviour described by Keynes (1936).

In a large part of HAMs in finance there are two types of agents that should represent the real financial market actors. The first group are the fundamentalists. They decide based on underlying economic factors. So they have some benchmark guess of the fundamental value and buy the undervalued assets while selling the overvalued ones. The second group is represented by the chartists. These represent the technically based traders. They decide solely by past movements of prices.

Amman *et al.* (2006) introduces the area of HAM by a simple model by Zeeman (1974) that illustrates the basics. Its purpose is to qualitatively describe

the stylized fact of the changing of bull and bear markets. It contains a lot of behavioural elements still used in HAMs. The model consists of fundamentalists and chartists as they were described earlier. In Zeeman (1974) model chartists are only trend followers simply copying the direction of the market. There are only three variables. The rate of growth of a stock market index, the share of chartists and the excess demand for stock of the fundamentalists. Zeeman shows that the simplest model that can be derived from his seven hypotheses based on qualitative features of the stock market and the behaviour of individual agents is the cusp catastrophe model with a slow feedback flow. In Amman *et al.* (2006) the reader can find a nice description of the model.

Brock *et al.* (2005) tries to describe the dynamics of behaviour of purely heterogeneous markets with a huge amount of trader types. They apply it to an evolutionary market and show that increase in the heterogeneity of agents and diversity of their beliefs may lead to unpredictable outcomes and excess volatility.

In Mueller & de Haan (2009) a simulation of market for new cars is described. The aim of this model is to forecast the effects of feebate systems based on energy labelling scheme. Consumers, the agents, have differing price elasticity and various reaction options to feebates. Consumers decide among few car sizes. The outcome of the model should serve to understand environmental and market effects of feebates. It is similar to our model as it features a market with differentiated products with a discrete choice.

Last years have brought extensive use of ABM in macroeconomics. Very ambitious projects such as EURACE model, a simulation of the European economy. As written in Deissenberg *et al.* (2008) is a complete model of all important markets. These include market for consumer goods, labour, credit or investment. It is unique among all ABMs as it contains a huge amount of agents. Not only it does simulate more than one market but it establishes a connection among the individual parts of the model. They stress out that the agent in EURACE acts within a context. That means not only he can interact on multiple markets but he can become a buyer on one market and seller on another. The model contains many classes that represent multiple types of agents. The connection between ABM and OOP is discussed later.

These large scale agent-based models can possibly be the future of economic modelling. They can be implemented with scalable and modular architecture. Various scenarios can be tested with only a small amount of work. It is a good example of using new technologies in economics and of an interdisciplinary

approach.

In Dawid *et al.* (2012) the assumptions and economic features of an EURACE extension Eurace@Unibi are explained. The model is extensively described and serves as an inspiration for our model. They try to base the behaviour of agents on empirically observed data. That is a good approach but gathering high quality empirical data is a hard task.

The time in the model flows in discrete steps in a manner similar to our model. In EURACE there are more types of time steps as there are more types of agents who can require a different schedule of activation. A day of activation is randomly chosen for every agent in the simulation and then preserved throughout the simulation. Agents often have more than one activation interval and are registered to listen to some important events. For instance a firm chooses its retail price every year and can react to a bankruptcy protocol. Other actions are taken every month. Households receive their income every month, perform their consumption-saving decisions and pay taxes. Every week they interact on the market for consumption goods. Unemployed citizens can react to events on the labour market. The decision rules are fairly sophisticated and based mostly on microeconomic theory and empirical data.

3.4 Spread and Justification of ABM

Despite the widening application of agent-based modelling in economics, research papers based on ABM fail to reach top economic journals. ABM somehow still lies on the edge of economic research. The simulation approach is often viewed as somehow inferior to the purely mathematic approach. Leombruni & Richiardi (2005) tackle these opinions showing that simulation approach is not less sound than the mathematical modelling. They build on the assumption that for every time index t the agent $i, i \in 1...n$ is well described by his state variable $x_{i,t}$. Then they let the evolution of the state variable in time be specified by a difference equation as:

$$x_{i,t+1} = f_i(x_{i,t}, x_{-i,t}, \alpha_i) \tag{3.1}$$

where α_i are parameters of the agents and by $x_{-i,t}$ are represented the states of all the other agents i.e. excluding the agent i. Then we are interested by the outcome or some macro feature that is typically a function of all the state variables. The denote this variable as Y. So in time t we have an equation:

$$Y_t = s(x_{i,t}, ...x_{n,t}) \tag{3.2}$$

Now they ask a question whether it is possible to solve this equation for each t in a general case. By using the theory of differential equations they show that the value of Y at any possible time t is uniquely determined by the initial conditions of the formerly defined system and the parameters α_i. They conclude that this formalization can be used to describe not only the ABM model but also the traditional dynamic micro framework.

Additionally, they show that the criticism of the difficulty of estimating ABMs fails. It is just necessary to use different estimation methods. Agent-based models are sometimes even advantageous because they enable us to model more complex phenomena and solve the model for a huge amount of data. The computation process is much more traceable as we can modify the model with ease.

3.5 Implementation of ABM and OOP

i The choice of implementation techonolgies seems to be often made depending on the skills of researchers involved. During smaller projects executed by e.g. economists a simple modelling environments such as NetLogo are often used (Tisue & Wilensky 2004). Simple simulation models are easy to design and an interesting visualization is available. That is very useful for models of one separate phenomenon. For large scale models such as EURACE these modelling environments are inappropriate. Huge computational complexity requires more sophisticated tools that operate on a lower level. Also modularity and ease of changing a part of the model is essential. Programmers are then required to write a simulation that will be both fast and maintainable in some standard programming language.

One of the most popular approaches for designing computer programs is called object-oriented programming. In fact this approach is very useful for writing agent-based simulations (Amman *et al.* 2006). The prototype of the agent can be naturally represented by a class in any OOP language. The individual agent is then an instance of the class. That means the agent has the same interface as every other agent. What is different is the data behind which determines the behaviour of the specific agent.

The object is in fact a container for data that allows actions to be taken

Table 3.1: OOP and ABM Connection

Object Oriented Programming	Agent-Based Modelling
Class	Agent prototype
Object	Agent
Interface	Set of possible actions
Method/Function	Individual type of action
Property	Data and Parameters

on its data. These actions are predefined and are called methods or functions. They can be parametrized on the data or even modify them (Szyperski 2002). The object oriented approach is used extensively in our model. The main advantage is high freedom of implementation. In addition it is easier to customize simulation written in a casual programming language. But as mentioned before for smaller simulations simplified programming environments are often sufficient and can save a non-trivial amount of time.

Figure 3.1: Example of an Agent in OOP

```
public class Agent
{
  // publicly accesible data
  public int Wealth{ get; set;}
  public int Age{ get; set;}

  // private data
  private Agent Neighbour { get; set;}

  // called when the object is constructed
  public Agent(Agent neighbour)
  {
    // initialization
    this.Neighbour = neighbour;
  }

  // notifies neigbour
  public void Notify()
  {
    // calls notify on the neighbour
    this.Neigbour.Notify();
  }
}
```

See Figure 3.1 for an example of a single agent represented by a C# class.

It consists of properties that represent data and of a method that shows the possibility of interaction among agents.

3.6 Validation of ABMs

An important part in the development of agent-based models is their validation. In Moss (2008) the author describes the possible methods of validation based on Windrum *et al.* (2007).

1. Indirect calibration begins with identifying stylised facts on the macro level. These can be some relations among macroeconomic variables such as GDP, growth or employment. Then we base the design of the model on empirical evidence from the real world. This is often based on microeconomic theory or on behavioural economics. Then the parameter space is restricted based on stylised facts or empirical knowledge. Sometimes Monte Carlo methods are used.

2. The Werker-Brenner calibration starts from the opposite side. At first, empirical facts are used for calibrating the ranges for parameters and find reasonable settings. Then we run a huge amount of simulations for all possible sets of parameters. The sets that are not in accord with the empirical facts are discarded. Then the range can be further restricted by some expertise in the underlying mechanics.

3. History-friendly calibration as the nomenclature suggests is based on including historical facts into the model. The model uses more empirics and less theory. The initial conditions and parameters are based preferably on historical evidence. The validation then proceeds by comparison of the outputs of the model with the historical facts.

Moss (2008) concludes that all these approaches are based on techniques and theories selected independently of the evidence. They are chosen before the design and implementation takes place.

Chapter 4

Model of the Software Market

According to our research no ABM model representing the software market has yet been made. We construct a new model based on the state of art in other ABM applications and the consumer decision theory. From Zhang & Zhang (2007) we take the well-reasoned concept of motivation. However, the decision rule they use, shows to be inappropriate in our model. The model in Zhang & Zhang (2007) is concentrated only on the study of the decoy effect. It is simple and has purely heterogeneous agents.

Ajzen (1991) inspired our model by its insights about the decision making process. It is one of the sources for the decision to include perception into the model.

Unlike standard economic models, agent-based models are generally represented not only by a set of equations but they contain other entities. It is essential to describe these entities first. The core of the model is the customer agent. His internal representation is the main theoretical part of the model. The behaviour is derived not only from economic theory, but we take into account other scientific fields such as psychology, sociology or marketing science. Said *et al.* (2001) describes interesting consumer theoretical framework. It comes out, that many concepts are bundled together by the marketing science. Scientists and practitioners that are concerned with marketing try to understand consumer behaviour without the methodological restraints of economics or sociology. Inspired by psychology they explore perception of the product and learning process.

Economics incorporates the view of a consumer as a rational being. The consumer is given a utility function he tries to maximize. The problem is that we are unable to measure utility and the full rationality of customer is often

too strong assumption.

Said *et al.* (2001) points to Antonides (1989) as an interesting attempt to combine the psychological theory of attitude with the economic theory of demand. After reviewing the individual approaches he tries to integrate the theories of utility maximization, theory of attitudes used in social psychology and the interesting theory of psychophysics. In psychophysics the relation of objective and perceived values are explored.

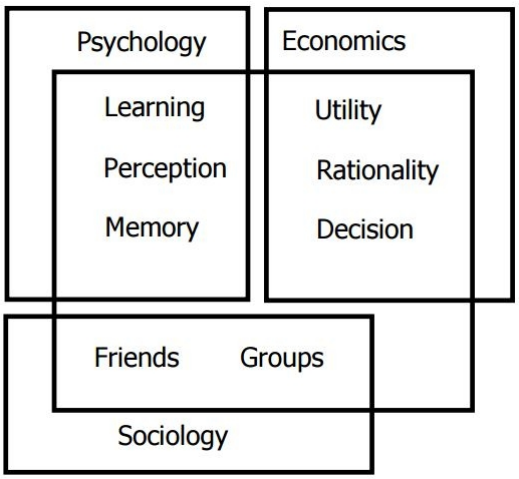


Figure 4.1: Theoretical basis of the model

Here we can see the advantage of the psychological attitude to customer on the micro-level. It can be used without the concept of utility and is very useful when the customer is not faced with a choice that would greatly affect his budget. When buying a car, the total budget and income play a huge role in the decision. While in the case of a much cheaper goods, these effects are diminished.

In our model we take into account these findings and on their theoretical basis we construct a reasonable representation of the software market. We make an analysis of the specifics of the market and find a suitable representation integrating together the economical, psychological and sociological perspectives.

4.1 Structure

Our model has two central types of entities. The first type is product and the second type is customer. We decided to represent the supply side only by the product. This is supported by the assumption of low variable and transaction

costs. To simplify the model and understand the market we choose to ignore the role of the company. We assume that the internal processes of the company do not play an important role in determining the structure of the market or market shares of individual products. Their modelling could be done using ABM or different methodology and integrated with the model of the market. In our model we simply take the company's decision as an input. Then we observe the reaction of the market. We can say that the model simulates the situation conditional on the product characteristics determined by the company.

4.2 Model Advantages

Most simulation models develop in time, but only a few of them take into account the individual memory and experience of each agent. In our assumptions in Chapter 2 we discussed their importance in understanding the processes on the software market.

We extend the concept of often simple memory parameters. In the model we implement perception memory based on Said *et al.* (2001). Additionally, memory of satisfaction with the product is included to represent the experience with the usage of the product. This should be a unique feature among agent based models and it is described later in the text with a greater specificity.

4.3 Customer Agents

Each agent is a representation of a particular customer. Agent's individuality is defined by a collection of parameters. In the model, customer is seen from a restricted perspective. We are not interested in his background but we are concerned only by the resulting characteristics that describe his behaviour and attitude on the software market.

Each customer that is a part of the model has the same mechanics of behaviour. What is specific to the individual are the decisions he makes based on his unique parameters. The model is populated by a chosen amount of agents. This is determined before the simulations starts and it is a part of the simulation settings.

4.3.1 Initialization

Some agent-based models are populated only with agents whose parameters are generated randomly from some standard distribution. In our model the parameters of agents are taken from normal distribution. However, for each group we can specify different mean and standard deviation. So there is heterogeneity caused not only by the underlying distributions from which parameters are taken. Differences among groups make a second type of heterogeneity. The appearance of the settings dialog can be found in Figure 4.2 on page 18.

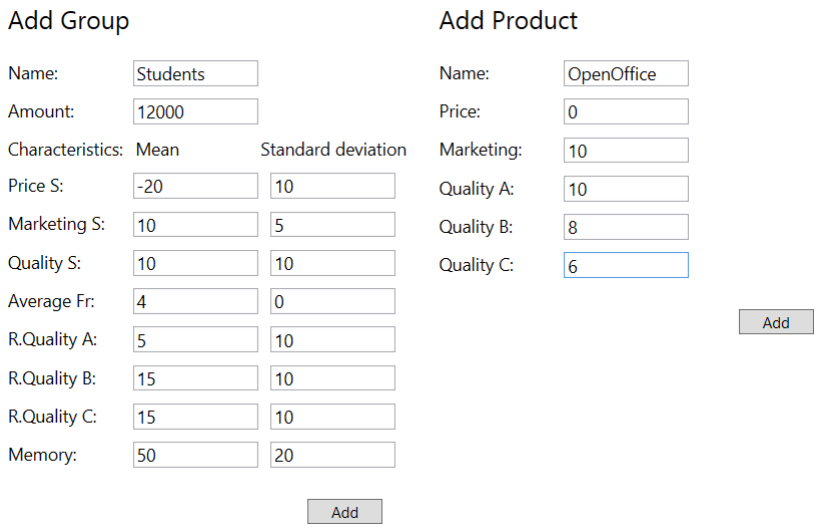


Figure 4.2: Simulation Settings Dialog

4.3.2 Parameters

The parameters of individual customer are described in Table 4.1 and the calibration of their effect is discussed later. The behaviour is based mostly on these parameters.

4.4 Agent Behaviour

We proceed with explaining the derivation of the behaviour of the customer agent. In the following text for the sake of simplicity we speak about one specific customer. All we derive for this customer, unless we explicitly state otherwise, applies to the customer agent in general.

Table 4.1: Parameters of the Customer Group

Group Parameters	Explanation
Price sensitivity	Price elasticity coefficient
Marketing sensitivity	Coefficient representing the intensity of perception
Quality sensitivity	Coefficient representing reaction to the quality level
Average friends	Average amount of friends of each group member
Required quality X	Required level of quality X of the product
Memory coefficient	Coefficient describing speed of forgetting
Amount	Number of customers of this group to be created

4.4.1 Motivation

The central question of this section is how the customer decides which product to buy. In Chapter 2 we assume our customer makes a decision that is partly rational. We admit that there is a certain level of unpredictability but his decision is correlated with the parameters. So we have to derive some measure of his willingness to buy the product. This characteristic will be called motivation and is inspired by motivation in Zhang & Zhang (2007). We will denote the motivation of the consumer to buy product i as M_i. The motivation is computed in every step of the simulation in which the customer wants to buy a product. It is obvious that the motivation depends on both the characteristics of the customer and the product. The motivation is the core of customer's decision so it depends on two variables described further in the text. The perception that represents a long-run view of a product even if the customer has no direct experience with its usage. We will denote the perception of the consumer of product i as P_i. Now we propose the following equation for the computation of the motivation of the customer at time step t:

$$M_{i,t} = p_i \times s_p + P_{i,t} \tag{4.1}$$

where by p_i we denote the price of product i at time t. Variable s_p represents the sensitivity of this particular customer to changes in price. Finally, $P_{i,t}$ denotes the perception of the consumer of product i at time step t.

We set the dependence of motivation on price as linear. Consult Chapter 2 for justification. The motivation is directly dependent on the perception of product i at the present time step t.

4.4.2 Perception

Now we have to explain what perception is and why have we decided to use it in our model. Perception is supposed to capture the continuous process of recognizing the product as an individual entity. The customer gradually gathers information about the product. That simulates the reality when the information customer gets is not complete and depends on the intensity with which he observes the state of the product and on the intensity of the marketing activity of each brand. Every piece of information the customer gets is subject to the process of degradation.Its importance decreases over time. That is strongly connected with how long and with what priority the customer uses his past experiences. So we introduce perception variable based on our assumptions stated in Chapter 2. We construct the following relationship for perception:

$$P_{i,t} = m \times P_{i,t-1} + \Delta P_{i,t} \tag{4.2}$$

where $P_{i,t}$ is the perception of product i at time step t. Parameter m is the memory rate. $P_{i,t-1}$ naturally denotes the perception at the last time step $t-1$. $\Delta P_{i,t}$ is the actual change of perception based on the parameters in time t. We propose the following equation for the perception change:

$$\Delta P_{i,t} = s_m \times (F_{i,t} + Q_{i,t}) + S_{i,t} \tag{4.3}$$

Parameter s_m represents the sensitivity of the particular customer to marketing activities of the products. It is the de facto rate of propagation of the product characteristics to the perception variable. $F_{i,t}$ is the influence of friends of our customer in time step t concerning product i. Every customer can have a set of friends that influence his attitude towards the product. This is discussed in Chapter 2. We have to choose a reasonable source of the influence value. Perception seems to be the best option as it is a synthesis of the friend's attitude towards the product. This brings to our model some kind of spatial interaction. We set a following relationship for the influence of friends:

$$F_{i,t} = w \times \sum_{f \in friends} perception_{i,f,t} \tag{4.4}$$

where w is the follower tendency parameter of our customer. It specifies how much does the customer take into account the opinion of his friends when rating the product. The second part is just a sum of actual perceptions of all

his friends.

Now we return to our equation (4.3) for the perception change. We have to explain what does $Q_{i,t}$ represent. It should be a measure of how much does the product meet the requirements. Justification of this measure is provided in Chapter 2. We choose two different relationships for the influence of the product requirements. When the product quality requirements are met the influence of an individual requirement is:

$$Q_{i,q} = q_{i,q} - r_q \tag{4.5}$$

On the contrary, when $q_{i,q} - r_q < 0$ the relationship we choose is:

$$Q_{i,q} = -(q_{i,q} - r_q) \tag{4.6}$$

where for both equations (4.5) and (4.6) $q_{i,q}$ is the quality q of product i. Parameter r_q is the quality requirement for quality parameter q of our customer. Thus we explored the relations connected with the individual quality parameter. Now in aggregate we have the following equation:

$$Q_{i,t} = s_q \times \sum_{q \in qualities} Q_{i,q} \tag{4.7}$$

So the value of $Q_{i,t}$ is nothing more than a sum of all the individual effects. We expect no strong synergic effect of the individual effects. Also the absence of more features is distinctively worse than absence of one. Unsatisfactory state of some quality does not cause complete ignorance of other qualities when making a comparison.

Until now we have ignored the last variable that has effect on the perception change. $S_{i,t}$ represents a third important characteristic computed for the customer agent. A separate explanation is presented.

4.4.3 Satisfaction

The satisfaction has mechanics similar to perception. It is computed by similar equations but it does not depend on marketing intensity and sensitivity. Also recommendations of friends are not included as when using the product personal experience dominates. Moreover, the recommendations are still included in the perception. An important feature is that satisfaction influences perception as was already mentioned. We have to emphasize that this makes the satisfaction

play role even after abandoning the product. The satisfaction of chosen product i at time step t is:

$$S_{i,t} = m \times S_{i,t-1} + \Delta S_{i,t} \tag{4.8}$$

where m is the memory parameter and $S_{i,t-1}$ is satisfaction with product i in the past step. The value of $\Delta S_{i,t}$ is

$$\Delta S_{i,t} = s_q \times \sum_{q \in qualities} Q_{i,q} \tag{4.9}$$

Parameter s_q is the level of quality sensitivity. The sum is identical to the one in equation 4.7. For other products than the one that is used by the customer only the process of forgetting is active:

$$S_{i,t} = m \times S_{i,t-1} \tag{4.10}$$

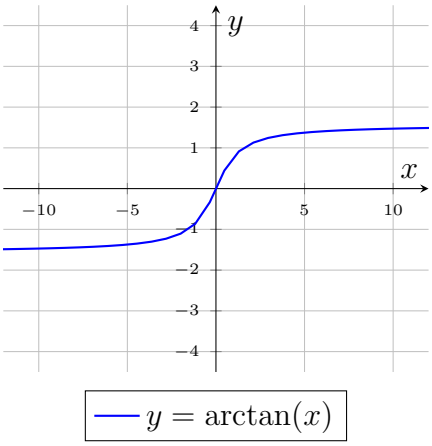
4.4.4 Willingness to Change

Not every customer wants to buy a new product in each step. But how to decide which one will choose a new product? In general, customers that are satisfied with the use of the product are less likely to change their choice. On the contrary, unsatisfied customers who find that the product they use has serious flaws or that it cannot satisfy their needs will probably try to choose a better suited option.

Now we see that some measure is needed to determine the probability the customer will make a new choice. This was one of the reasons why we introduced satisfaction variable earlier in the text. It captures the long-run experience of the customer with the usage of the product. So we have the measure we needed, but additionally we would like the probability to reflect the behaviour of completely satisfied or strongly unsatisfied customers.

We need a function $f(S_{i,t})$ whose value around 0 will be some chosen default probability p. Then in the negative domain we want the value to degrade from p. We would like the function to be convex as the importance of the level of satisfaction lowers with distance from 0. In positive domain we want the value to rise similarly as it goes down in the negative domain. Naturally, we want the values to lie in $[0, 1]$ interval. After exploring possible options we decide to use $arctan(x)$ as $f(S_{i,t})$ because it has some of the desired features described above.

Figure 4.3: Graph of $arctan(x)$



On the figure above we can see several problems. Firstly, the range of values is $(-\frac{\pi}{2}, \frac{\pi}{2})$. Also it does not logically correspond to our standard values of satisfaction. So we have to make a transformation of $arctan(x)$ to get:

$$f(S_{i,t}) = \frac{\arctan(\frac{x}{c}) + \frac{\pi}{2}}{3.5} \tag{4.11}$$

We add $\frac{\pi}{2}$ to move the range of values above zero and then divide the sum by 3.5. This is done in order to constrain the values into the $(0, 1)$ interval. A value larger than π is chosen to leave some space for making a new choice when satisfied. A reasonable coefficient c has to be chosen and that is discussed later in the text.

So now concerning the probability that the customer i will not make a new choice in step t we propose the following equation:

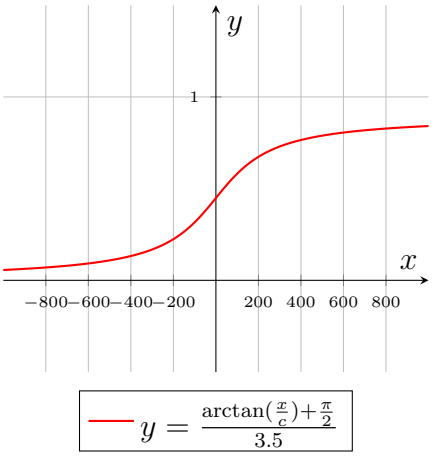
$$P_{i,t} = f(S_{i,t}) \tag{4.12}$$

This is applied at each step for each individual customer. A random number from the $U(0, 1)$ is drawn and if it is higher than $P_{i,t}$ the customer makes a new choice.

4.4.5 Decision

Now we analyze the decision of the customer. Earlier in the text we explained the motivation variable $M_{i,t}$. It is computed in each step and serves as some kind of measure of actually perceived utility inherent in buying product i. So we have some measure of utility from buying product i for each $i = 1, ..., n$ where n is the number of products. We then base our decision on the sequence

Figure 4.4: Graph of $f(S_{i,t})$



of all the motivations. In Zhang & Zhang (2007) they use a very simple decision rule:

$$MAX(M_1, M_2, ..., M_n) \tag{4.13}$$

This rule is elegant but it does not leave any space for unobserved factors. It assumes complete rationality of the customer. We are searching for some partial function that is easy to understand and leaves some space for randomness of the decision. A core requirement is that higher motivation for buying product i results in higher probability that it will happen so. Based on this requirement we propose the partial function $d : \mathbb{R}^n \to 1, ..., n$ defined by property:

$$P(M_i) = \frac{M_i}{\sum_{p=1}^{n} M_p} \tag{4.14}$$

Thus the probability of choosing product i is based on the relative value of the motivation to buy it to the total sum of motivations. This partial function satisfies the property that $\sum_{p=1}^{n} P(M_p) = 1$. Customer chooses the product determined by the function $d(M_1, M_2, ..., M_n)$. Any modification of this function that best suits the particular case can be used.

4.5 Product

Another type of entity in our model is product. The product is not an agent as the customer entity. It represents the supply side of the market. A data about income and market shares of the individual products can be collected.

Figure 4.5: Simulation Diagram

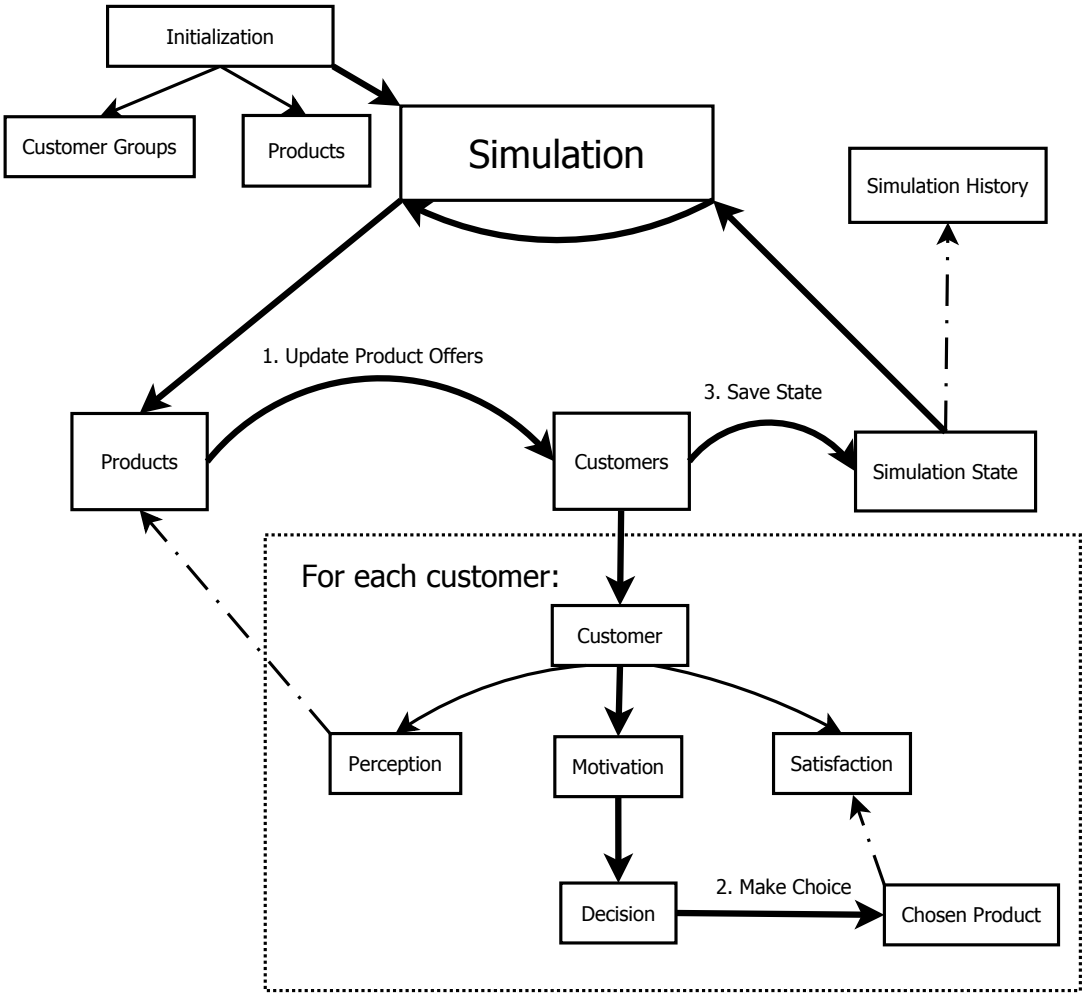


Table 4.2: Product Parameters

Product Input Parameters	Product Output Parameters
Price	Income
Marketing	Market Share
Quality Parameters	Product History

4.6 The Big Picture

The overall structure of the model is shown by the diagram in Figure 4.5. At first, the model has to be initialized with starting settings of customer groups and available products. In each step of the simulation exogenous modifications of the environment can be made. They are applied before the simulation step takes place. Product offers are updated and sent to the customers. After that each customer makes a sequence of actions. These are put together in Table 4.4.

Table 4.3: Simulation Actions

Simulation Action	Order	Remarks
Update	1)	Apply exogenous changes
Call All Customers	2)	Customers decisions are resolved
Save State	3)	Save all variables of current step

Table 4.4: Customer Actions

Customer Action	Order	Remarks
Compute Perception	1)	Computes and saves perception
Compute Motivation	2)	Computes motivation
Make New Choice?	3)	If true, call 4)
Choose Product	4)	Called only if 3) is true
Compute Satisfaction	5)	Computes and saves satisfaction

Customer starts with computing his perceptions of all the products and saves them. Then he computes his motivation which will be used for the current step. Then the customer decides based on his satisfaction whether he will make a new choice. If that is the case he makes a new choice and finally his satisfaction is computed based on the product currently in use. Then the state of the simulation is saved, see Table 4.3.

Chapter 5

Implementation and Model Analysis

To support the theoretical description of our model in Chapter 4 we introduce a simulation that implements the theoretical framework of the model.

It is written in C#, an object-oriented programming language, and runs on the .NET framework virtual machine. A visual interface of our simulation was developed using WPF (Windows Presentation Foundation) technologies. It can produce interactive visual output, enables customizing the parameters of the model and modifications during the simulation. Export of simulation data to CSV format is possible.

In this chapter we describe the interface of our application to enable readers running their own simulation. In the second part we prepare a test scenario and test effects of various modifications on the market shares of the products on the software market.

5.1 Model Application Interface

The application consists of the simulation and three separate windows in Figures 5.1, 5.2 and 5.3. First window is dedicated to setting the parameters of the simulation. The second one serves as a control panel and enables modification of the product during the simulation. The last one shows output product data and a graph of market shares.

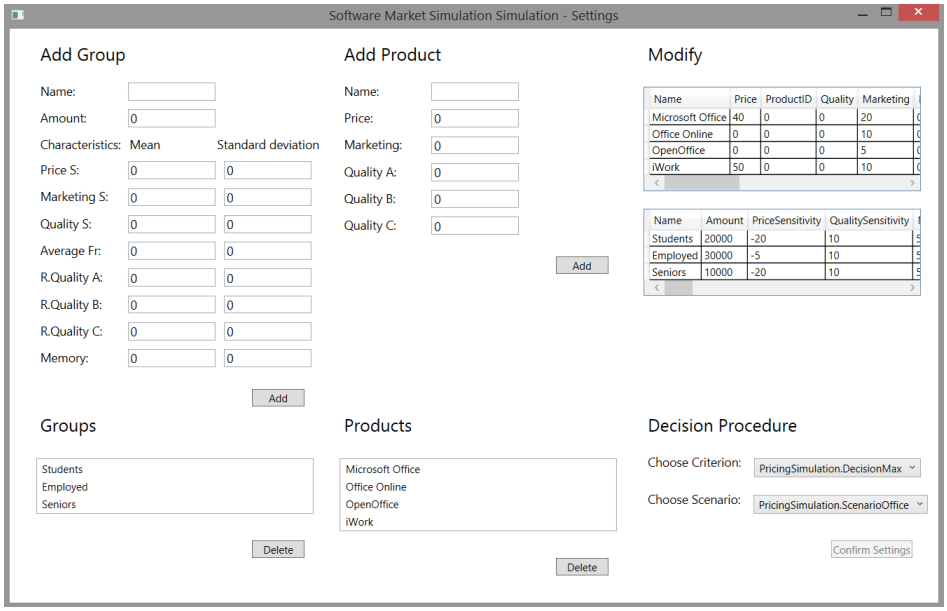


Figure 5.1: Settings Window

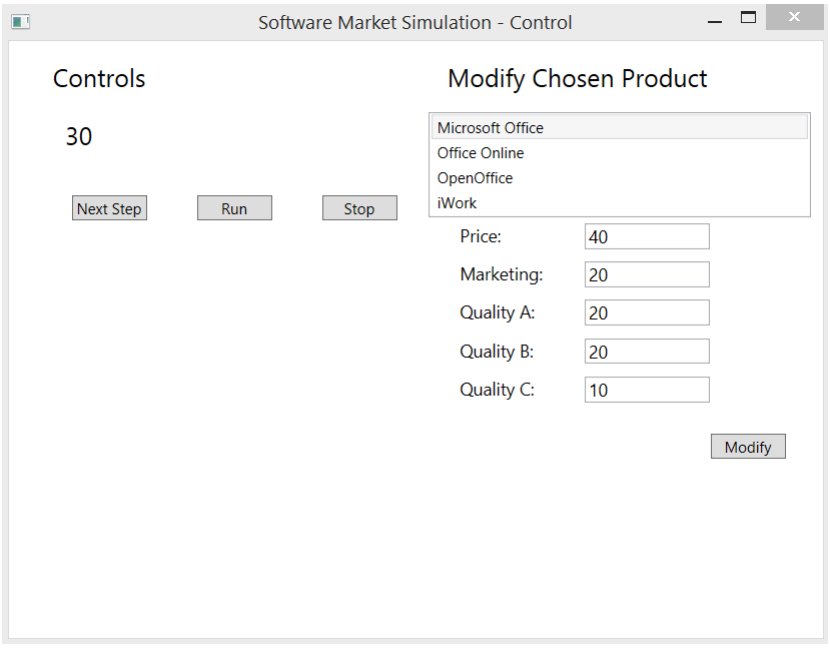


Figure 5.2: Control Window

In the settings window complete configuration of the simulation is available. Aside from the possibility of adding new products and customer groups the user of the application can remove them or modify existing settings. One of the predefined decision rules can be chosen. Other option is to write a new

decision rule and encapsulate it in a form of C# class derived from Decision class. In a similar way various predefined scenarios can be constructed and then chosen from the settings window.

The control window enables the user to run the simulation. It can be run in a step by step mode or in an automatic mode. In the control window modification of an individual product is enabled. An indicator of the current step number is shown.

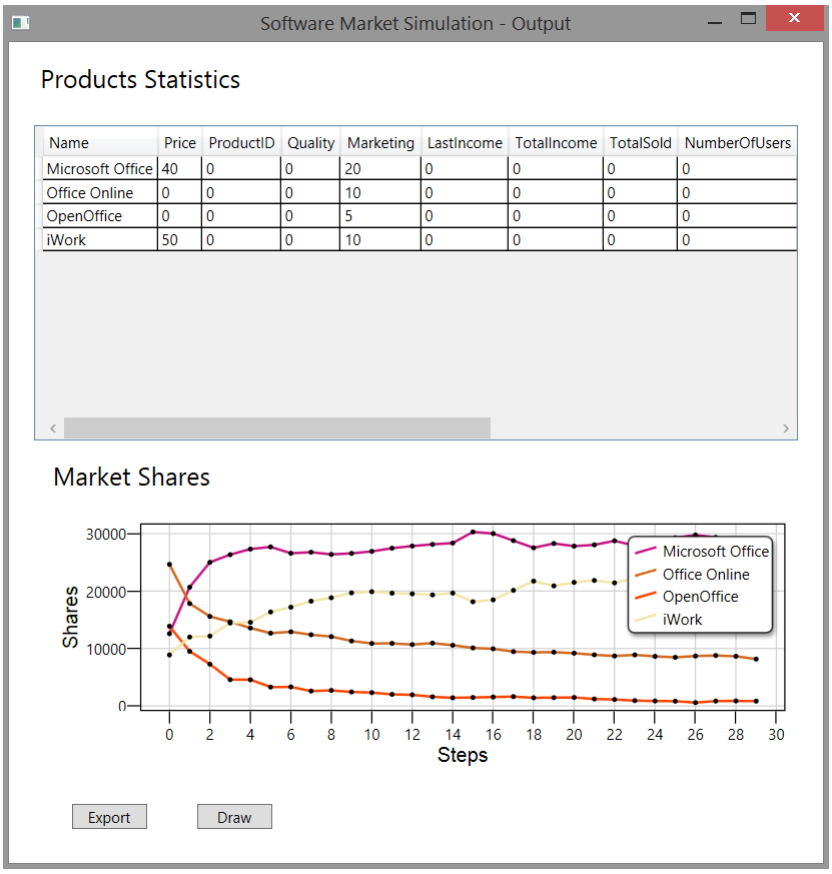


Figure 5.3: Output Window

In the output window there is a list box where all the products are listed. Their properties are displayed and can be modified at will. At the bottom, there is a graph that is capable of displaying the market shares of the individual products. There is a button that enables redrawing the graph. Using the two remaining buttons complete data about products and customers can be exported to CSV and saved to a chosen location. See Figure 5.4.

Figure 5.4: Sample of Exported CSV Product Data

```
STEPNUMBER,PRODUCTNAME,NUMBEROFUSERS,LASTINCOME,INCOME,SOLD,
1,ProductA,3,15,30,3,
1,ProductB,0,0,0,0,
2,ProductA,3,0,30,3,
2,ProductB,0,0,0,0,
3,ProductA,1,0,30,3,
3,ProductB,2,6,12,2,
```

5.2 Calibrating the Model

The important part of using the model is its calibration. As a most suitable calibration strategy we decide to use indirect calibration described in Chapter 3. We have derived our rules from stylised facts from both the macro and micro level. This derivation is explained in the preceding text. Then the indirect calibration approach guides us to restrict the parameter space only for reasonable parameters. We need to scale the parameters to a reasonable domain. Another requirement is that the proportions among the parameters should be in accord with the reality.

Table 5.1: Reasonable Values of Customer Parameters

Cutomer Parameter	Range of Mean
Price Sensitivity	$[-40, 0]$
Marketing Sensitivity	$[0, 20]$
Quality Sensitivity	$[0, 20]$
Average Friends	$[0, 10]$
Quality Requirement X	$[0, 20]$
Follower Tendency	$[0, 100]$
Memory	$[0, 100]$

So for setting up the parameters we should take the empirical data and transform them to our parameter space. For the sake of simplicity all computations in our model are done with integers. After some tests and analysis of the equations we found a reasonable range of individual parameters. In Table 5.1 and Table 5.2 a reasonable range of individual parameters can be found. The relative proportions of individual values depend on the endogenous input and the parameters should be scaled with a constant proportion to mostly fit in the intervals from the tables.

Table 5.2: Reasonable Values of Product Parameters

Product Parameter	Range of Mean
Price	$[0, 50]$
Marketing	$[0, 20]$
Quality X	$[0, 20]$

Some of the parameters can be generated from the normal distribution to simulate a heterogeneous population. The values are taken from $N(\mu, \frac{\sigma}{10})$ where μ is the mean that is set in the settings window and σ is the scaled standard deviation inserted by the user. The only reason for this scaling is to enable more precise setting of σ.

5.3 Observing the Effects

To demonstrate that the model behavior we test effects of various modifications and scenarios. We encourage the reader to do the same (follow this link for a zip archive with the simulation `https://www.dropbox.com/s/l5sc8kucwhz01h9/Simulation.zip?dl=0`). We test how our model reacts to dynamic changes of different parameters of the product. The graph of market shares and its evolution after the modifications are observed. All the graphs in this section are the output of the application.

Table 5.3: Settings of the Test Scenario - Customer Groups

Mean (Std. Deviation)	Students	Professionals	Retired
Amount	5000	5000	3000
Price Sensitivity	-10 (20)	-10 (20)	-20(20)
Marketing Sensitivity	10 (10)	10 (10)	10(10)
Quality Sensitivity	5(30)	10 (30)	10(20)
Average Friends	5	5	3
Quality Requirement A	10(40)	20(20)	5(20)
Quality Requirement B	10(40)	20(20)	5(20)
Quality Requirement C	10(40)	10(20)	10(20)
Follower Tendency	30	15	50
Memory	50	80	80

We have prepared a simple scenario that is used in the following tests. There are three products in the scenario together with three customer groups constituting a population total of 13.000. In Table 5.3 you can find the cus-

tomer groups settings of our scenario. The products are described in Table 5.4. This scenario is predefined and can be run in the application immediately. In the following sections we compare the individual of price, marketing level and quality modifications. For their comparison see end of this chapter.

Table 5.4: Settings of the Test Scenario - Products

Mean	Product A	Product B	Product C
Price	20	5	20
Marketing	20	10	10
Quality A	15	5	15
Quality B	10	10	10
Quality C	5	5	10

5.3.1 Price Effect

At first we study the effect of price. We leave the scenario unmodified and run the simulation for 10 steps. We see that the products hold a similar share of the market. Now, we decrease the price of Product A from 20 to 5. On the graph we can observe a gradual increase in the number of users of product A. Part of the customers who decide to make a new choice abandon their old product and buy Product A instead.

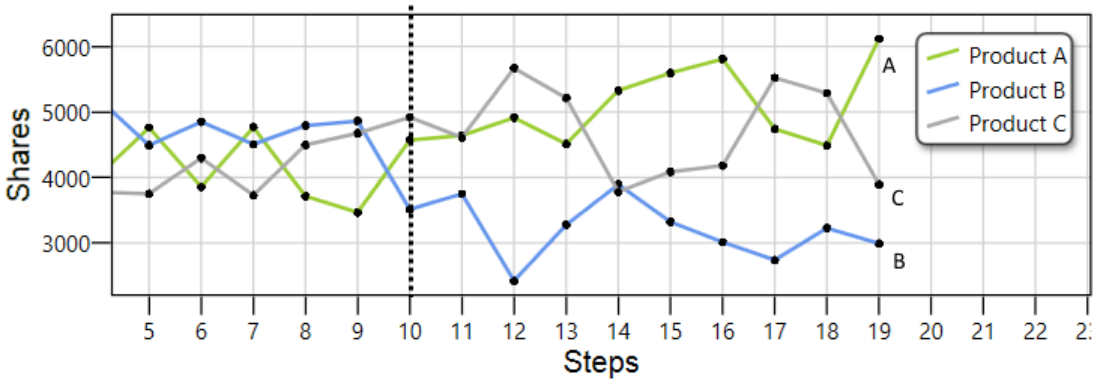


Figure 5.5: Price Effect - Decrease

Then we test the effect of price increase on the market shares. As before we let the simulation run in the original settings until step 10. Then we set the price of Product B to 15 from the former value of 5. We observe similar, but more dramatic decrease in market shares. The product which was inferior in

quality to the other two products got to their price level. The customers who bought this product because it was significantly cheaper consequently bought a different product.

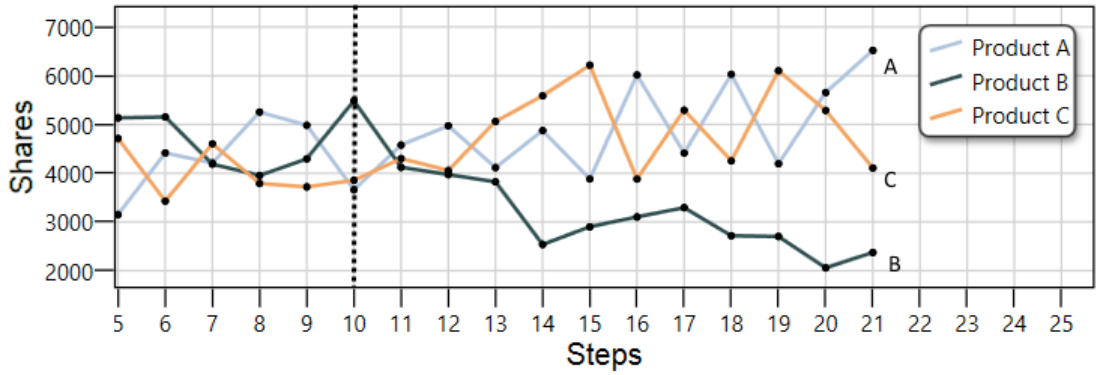


Figure 5.6: Price Effect - Increase

5.3.2 Marketing Effect

Now let us continue with the situaton after the price increase. We suddenly decrease the level of marketing of Product A at time step 21. The level of perception decreases over few time steps and is followed by decrease of the market share of Product A. The decrease in the level of marketing is from the value of 20 to 5.

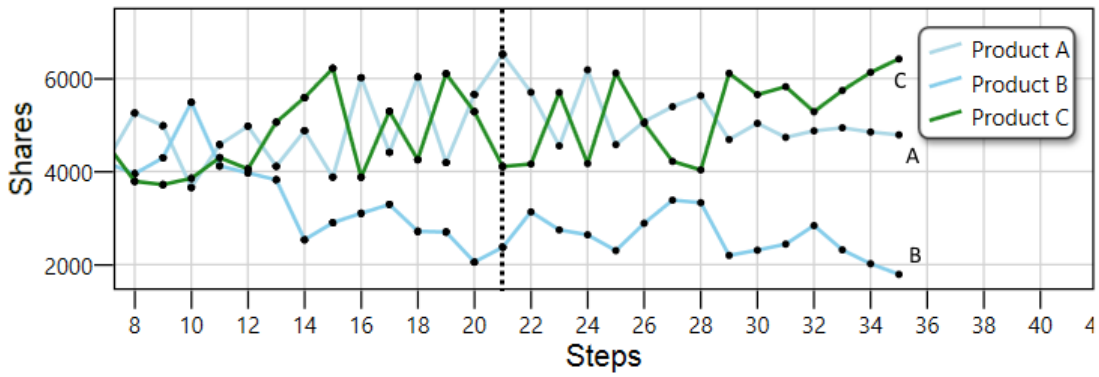


Figure 5.7: Marketing Effect - Decrease

In the second modification we run the scenario again and make the same decrease in the level of marketing of Product A as before. The difference is that now we apply the change before the simulation is run. We can observe

that the lower marketing level prevents the product from reaching the same level of market share as it had before. The interesting thing in this case is that the potential customers of Product A are taken over by Product C. So they migrate to the product with similar qualities because they just do not have enough information about product A. Still they do not choose Product B as it does not go along with their quality requirements.

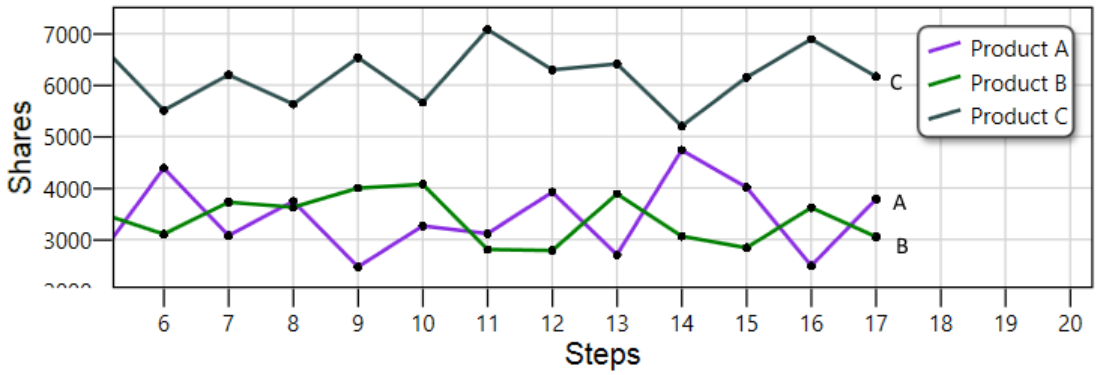


Figure 5.8: Marketing Effect - Start Decrease

5.3.3 Quality Effect

In this simulation run we have left the scenario unmodified until step 15. Then we decreased the quality A of Product C from 15 to 5. It took some time for the customers to perceive the change and to realize their unsatisfaction with the product. For a few steps the market share even increased following the former trend. But then a dramatic decrease in market share appeared thanks to strong unsatisfaction of former users.

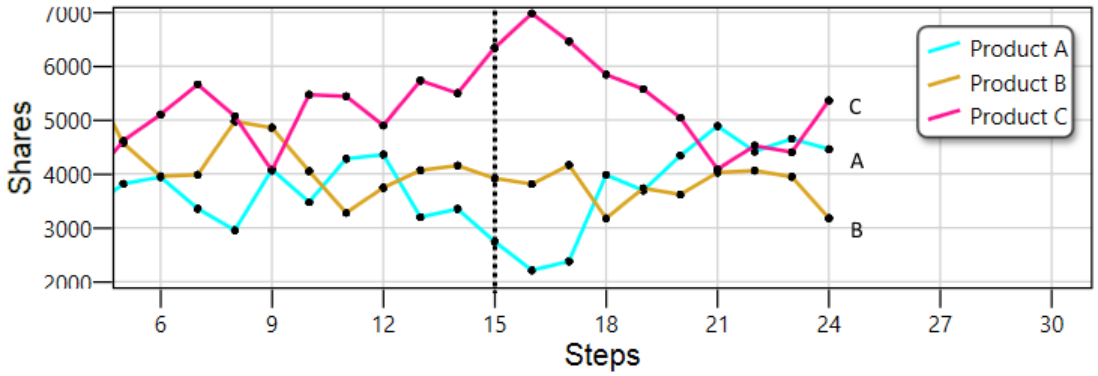


Figure 5.9: Quality Effect - Decrease

Contrary to the example before we test the effect of increase in quality. We choose a quality A of Product A that already has a satisfactory value of 15 and we set it to 25 at step 15. We can see that increasing an already high quality does not bring a large effect on customers. It just mildly increases the level of satisfaction and that is connected with a small increase in market share.

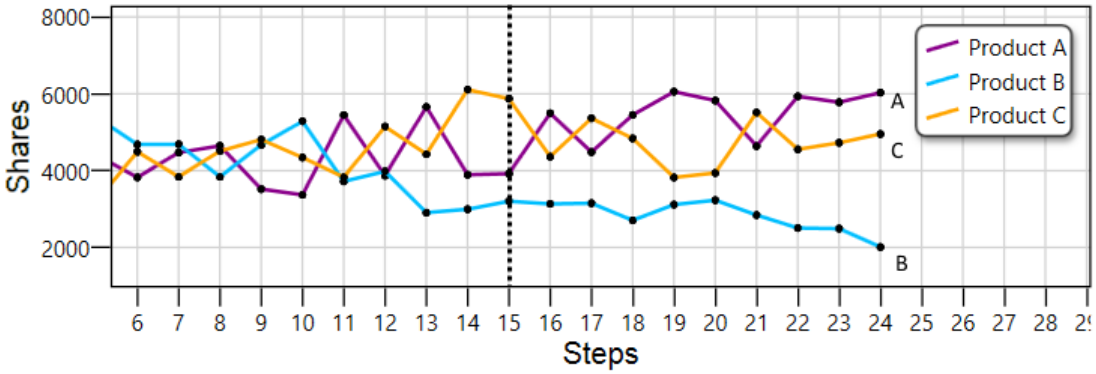


Figure 5.10: Quality Effect - Increase

5.3.4 Combined Effect

In this section we test changing multiple parameters at once and observe the effect. At first, we modify quality and marketing level at once. This time we take Product B which is inferior in quality to the other two products. We set all its qualities to 10 while decreasing level of marketing. This can answer the question if increasing quality at the cost of weaker marketing activities is profitable in this situation. The modification is made at step 10. We observe a small growth in market share of Product B initially. This is likely to be cause by the greater level of satisfaction with the product. But then the information about the quality change fails to reach the customers and a decrease in market share follows.

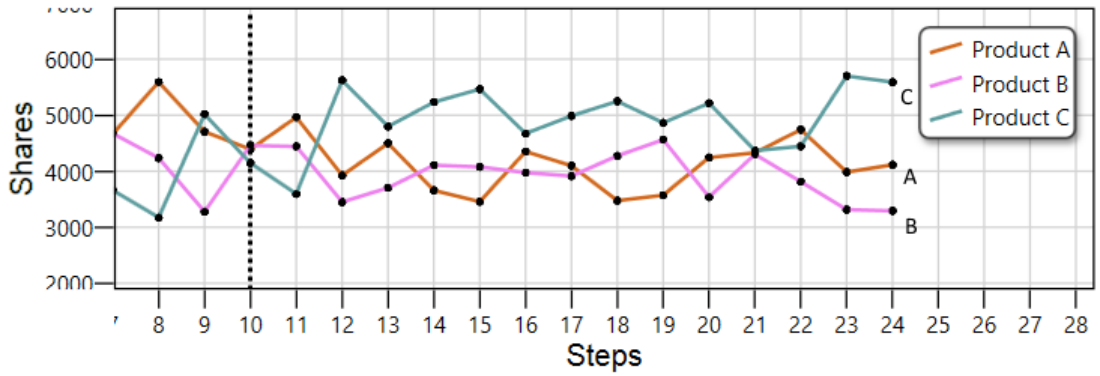


Figure 5.11: Combined Effect - Marketing Decrease, Quality Increase

5.3.5 Price Wars

Here we test the effects of multiple modifications in price simulating price wars between individul brands. Again starting at step 10 we decrease the price of Product A to 10. At step 13 Product C reacts with the same decrease of price to 10. At step 16 Product B decides to increase quality C to 15. This makes no significant change ont the market. Product C sets its price to 0 at step 19. We observe sharp reactions to the price changes in Product A and Product B with a new convergence after agreein on price value of 10. The price change in Product B to zero is not enough for the customers of Product A and C to stop using them.

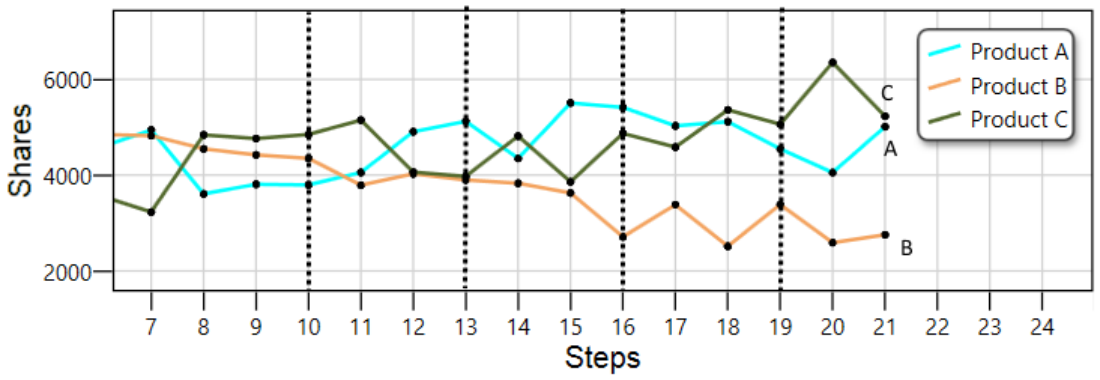


Figure 5.12: Price Wars

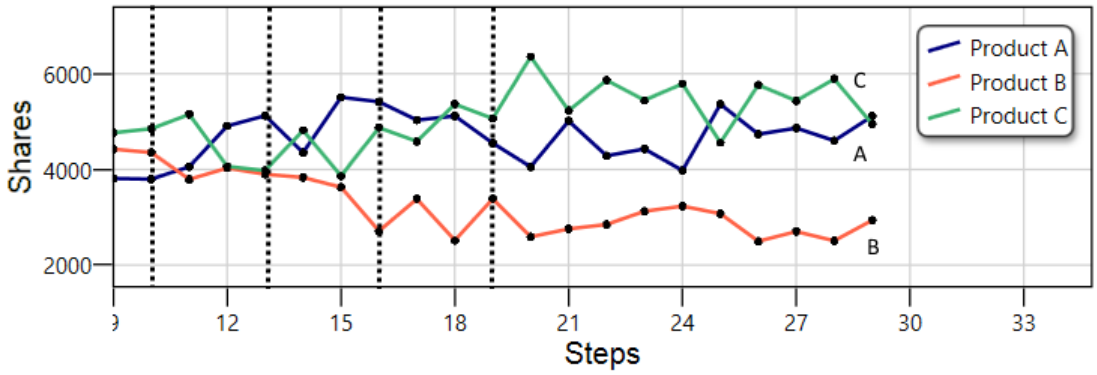


Figure 5.13: Price Wars with Additional Steps

Reducing price of the inferior Product B to 0 did not have a significant effect on its sales. Now we test the effect of giving the high quality product for free. At step 15 we decrease the price of Product A to 10 and at step 20 Product C reacts with becoming free of charge. After step 15 Product A gets edge over Product C. But the reaction of Product C is followed by strengthening their market share. However, it seems that most of the former users of Product A stick to their choice as they are satisfied and accustomed to their product. The viability of the free product strategy depends on the price sensitivity and willingness to change of the customers.

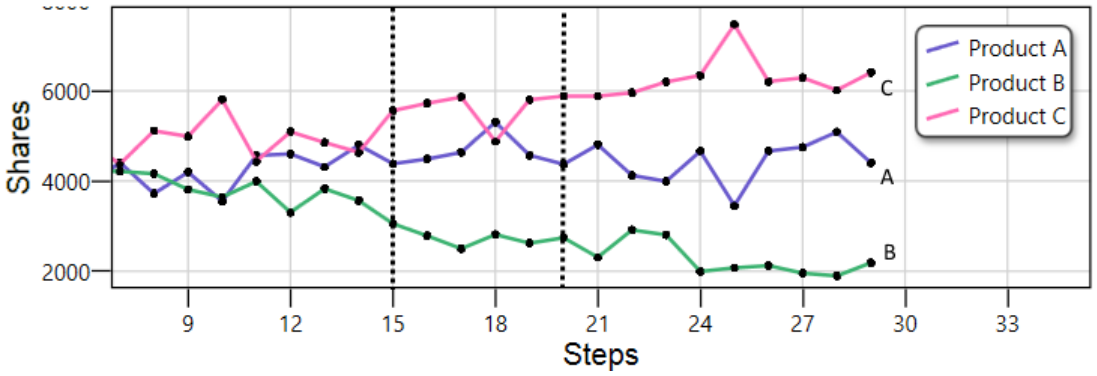


Figure 5.14: Price Wars 2

The effect of price modifications is predictable, but the level of satisfaction seems to have effect on the magnitude of their influences. Improving quality of a product has positive influence on sales, but its significance is higher if the product outreaches its competitor by this improvment. Decreasing quality works in an opposite direction but additionally significant lack of quality can

have a dramatic negative effect. Level of marketing influences the speed of reflection of the modifications. In Chapter 6 we present a small scenario with five products where these effects can be also observed.

Chapter 6

Case Scenario - Game Engines

In this chapter we use our application for simulating the market for game engines. Game engines are complex software products, that provide framework for development of computer games. Recently, professional game engines, formerly hardly affordable, lowered their prices drastically. Hollister (2014) can serve as an example of such an important price movement. Pearson (2014) is concerned by the competitive changes in pricing on the game engine market.

Table 6.1: Settings of the Game Engines Scenario - Customer Groups

Mean (Std. Deviation)	Students	2D Dev.	3D Dev.	Hobby Dev.
Amount	5000	5000	10000	3000
Price Sensitivity	-10 (20)	-10 (10)	-10(10)	-20(10)
Marketing Sensitivity	10 (20)	10 (20)	10(20)	10(10)
Quality Sensitivity	5(30)	15 (30)	15(30)	10(30)
Average Friends	3	5	5	1
3D Quality Req.	10(40)	0(40)	15(40)	5(40)
2D Quality Req.	10(40)	20(40)	10(40)	10(40)
Ease of Use Req.	5(40)	5(40)	0(40)	15(40)
Follower Tendency	30	15	10	10
Memory	40	50	70	40

We have chosen the settings trying to capture the situation on the market before the changes. According to our observations of the market we determined the parameters based on our perception of the individual brands. The goal of this scenario is not to simulate precisely the market for game engines but rather to show that such a simulation can be useful and that it can give us an interesting insight about the market. For the individual products we set their price, marketing level and three distinctive qualities. Quality of 3D development

tools, quality of 2D development tools and ease of use. Unreal Engine and Cry Engine are 3D engines aiming at professional developers while Unity 3D tries to be accessible by independent or hobby developers. Game Maker and Consruct2 are much more simple engines for making 2D games. The scenario settings are available in the application and we encourage the reader to try it. For more information consult Chapter 5.

Table 6.2: Settings of the Game Engines Scenario - Products

Mean	Unreal Engine	Unity 3D	Cry Engine	Game Maker	Construct 2
Price	50	30	30	10	0
Marketing	10	20	15	0	5
3D Quality	20	15	18	0	0
2D Quality	10	10	10	15	10
Ease of Use	0	10	0	15	20

At first we run the simulation with the inititial setting in Tables 6.2 and 6.1. We observe the long run situation on a static market. A graph of this situation can be found in Figure 6.1.

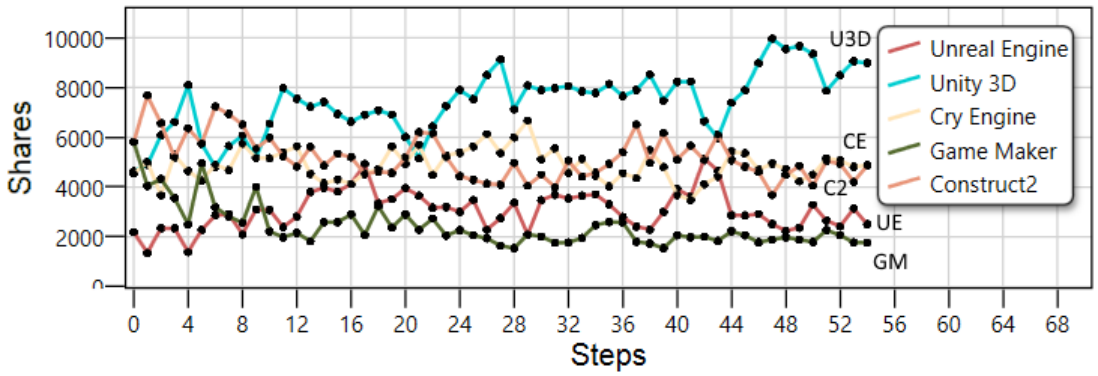


Figure 6.1: Game Engines Scenario - No modifications

Unity has the largest market share with a rising trend. This is a classical example of higher satisfaction with the product. Users of Unity are less likely to look for a new product a even then they are likely to stick with their former choice. On the contrary, part of the users of Cry Engine and Unreal Engine slowly migrate to Unity.

Now we simulate some dynamics on the market. We start with the initial settings above as before. At the time step 10 Unreal Engine decides to decrease the price from 50 down to 25 in order to become accessible to more customers.

In the meantime developers fom Unity work on better 3D functionality and in step 15 they develop a new version. Unity's 3D quality increases from 15 to 18. Then in step 20 both Unity and Unreal Engine decide to give their basic versions for free, so their price lowers to zero. The graph can be found in Figure 6.2.

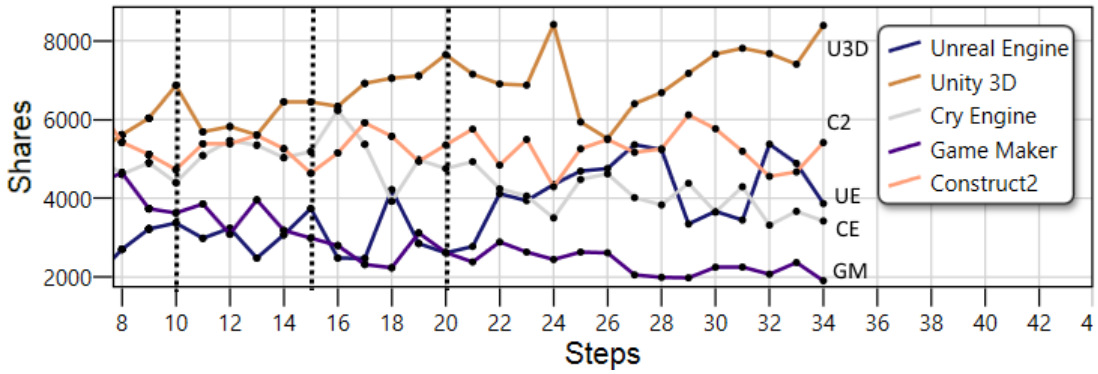


Figure 6.2: Game Engines Scenario - Dynamic Modifications

After the first price change of Unreal Engine there is only a little reaction of the market. After the increase of Unity's quality a slight increase in market share can be observed. At step 20 when both Unity and Unreal Engine decrease their price to 0 we can see a rise of Unreal Engine reaching a new level which is persistent. After another few steps a gradual increase in market share of Unity begins.

In the second scenario we decrease the price of prevalent product Unity at step 10 to 0. The other 3D engines react at turn 20. Unreal Engine reduces the price to zero and Cry Engine to half of the original value. The graph of this scenario is in Figure 6.3.

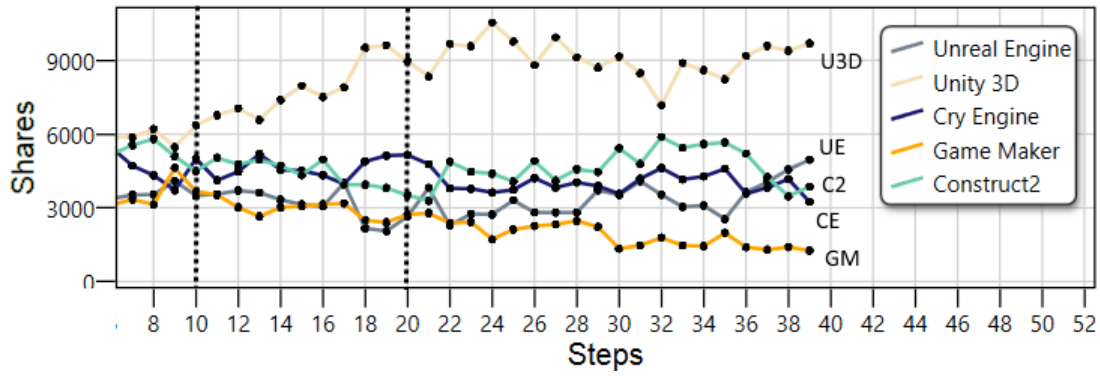


Figure 6.3: Game Engines Scenario - Dynamic Modifications 2

Following the price decrease of Unity we observe a predictable and stable rise in its market share. After the reaction of its competitors at step 20 Unity manages to hold its market share. This can be the result of high quality of Unity. Its users do not look for a new product so a majority of them would not even perceive the changes in price.

This scenario shows that there are interesting possibilities to utilize models of software markets. With more complex model utilizing empirical data the companies could test the effects of their pricing and development decision in a fast and effective manner.

Chapter 7

Conclusion

In this bachelor thesis we have applied agent-based modelling methodology on the software market. Specific features of the software market were discovered. We used them as a foundation of the logic when constructing our original model. After reviewing the literature concerning the ABM methodology we discussed the theoretical basis of the model. The model is constructed as a discrete simulation that is run in time steps. By following predefined rules the customer agents decide what actions should they take. They use a tripartite decision system based on three variables, that are computed by each agent for each available product. Motivation describes the actual willingness to buy the product, perception captures the long run view of the product and the satifaction describes the experience with usage of the product. The decision to buy a new product depends directly on the satisfaction. This decision system is an original contribution of this thesis and is based on theoretical concepts from Ajzen (1991) and Said *et al.* (2001). Its description can be found in Chapter 4. In the last part of the text we implement a simulation based on our theoretical model and describe its usage and calibration. It shows that the model gives reasonable output and can be used as a foundation for exploring the mechanics of the software market. The scenario concerning the market for game engines can be taken as a simple example of possible applications of the model.

The motivation to write this text was to understand the often marginalized software market and induce more interest into its research. The main contribution of the work lies in constructing and implementing an original model of the software market. It gives a solid framework for exploring the behaviour and processes of the individual actors and their macro outcome. The work shows how an agent-based model of a specific entity can be built from the ground.

The model we developed in the thesis does not have the ambition to serve as a complete representation of the software market. A larger team and various data with high quality are required to carry out a simulation with precise quantitative output. The model presented can be improved in several ways. The individual rules can be tested and compared with empirical data to validate them or to find more suitable rules. The simulation application itself has the potential to be greatly extended both in the sense of adding more options and computing a detailed and more useful output.

A natural extension of our work can be created using empirical data for calibration. The usefulness of the model is directly dependent on the amount and quality of the data about the customers and the products. The model can give us a reasonable output only for data that realistically represent the software market. With complex models based on empirical data companies could evaluate their potential pricing and development decisions. Thanks to the nature of computer simulations this can be done effectively and with low variable cost. We hope that our endeavour could inspire more interest in research of the software market. Possibly ABM methodology can be used for its analysis or even an extension of our model can be constructed.

Bibliography

Ajzen, I. (1991): "The theory of planned behavior." *Organizational behavior and human decision processes* **50(2)**: pp. 179–211.

Amman, H. M., L. Tesfatsion, K. L. Judd, D. A. Kendrick, & J. Rust (2006): *Handbook of computational economics*, volume 2. Elsevier.

Antonides, G. (1989): "An attempt at integration of economic and psychological theories of consumption." *Journal of Economic Psychology* **10(1)**: pp. 77–99.

Auchincloss, A. H., R. L. Riolo, D. G. Brown, J. Cook, & A. V. D. Roux (2011): "An agent-based model of income inequalities in diet in the context of residential segregation." *American journal of preventive medicine* **40(3)**: pp. 303–311.

Axelrod, R. (1997): "Advancing the art of simulation in the social sciences." In "Simulating social phenomena," pp. 21–40. Springer.

Bouchaud, J.-P. (2008): "Economics needs a scientific revolution." *Nature* **455(7217)**: pp. 1181–1181.

Brock, W. A., C. H. Hommes, & F. O. Wagener (2005): "Evolutionary dynamics in markets with many trader types." *Journal of Mathematical Economics* **41(1)**: pp. 7–42.

Castella, J.-C., T. N. Trung, & S. Boissau (2005): "Participatory simulation of land-use changes in the northern mountains of vietnam: the combined use of an agent-based model, a role-playing game, and a geographic information system." *Ecology and Society* **10(1)**: p. 27.

Dawid, H., S. Gemkow, P. Harting, S. Van der Hoog, & M. Neugart (2012): "The eurace@ unibi model: An agent-based macroeconomic model for economic policy analysis." .

Deissenberg, C., S. Van Der Hoog, & H. Dawid (2008): "Eurace: A massively parallel agent-based model of the european economy." *Applied Mathematics and Computation* **204(2)**: pp. 541–552.

Epstein, J. M. (1999): "Agent-based computational models and generative social science." *Generative Social Science: Studies in Agent-Based Computational Modeling* **4(5)**: pp. 4–46.

Epstein, J. M. (2009): "Modelling to contain pandemics." *Nature* **460(7256)**: pp. 687–687.

Epstein, J. M. & R. Axtell (1996): *Growing artificial societies: social science from the bottom up*. Brookings Institution Press.

Farmer, J. D. & D. Foley (2009): "The economy needs agent-based modelling." *Nature* **460(7256)**: pp. 685–686.

Hollister, S. (2014): "Epic drastically drops the price of unreal engine game development." `http://www.theverge.com/gaming/2014/3/19/5526086/epic-drastically-drops-the-price-of-unreal-engine-game-development`. Accessed: 2015-05-07.

Irwin, E. G. & J. Geoghegan (2001): "Theory, data, methods: developing spatially explicit economic models of land use change." *Agriculture, Ecosystems & Environment* **85(1)**: pp. 7–24.

Keynes, J. M. (1936): *The general theory of interest, employment and money*. London: Macmillan.

Leombruni, R. & M. Richiardi (2005): "Why are economists sceptical about agent-based simulations?" *Physica A: Statistical Mechanics and its Applications* **355(1)**: pp. 103–109.

Macal, C. M. & M. J. North (2010): "Tutorial on agent-based modelling and simulation." *Journal of simulation* **4(3)**: pp. 151–162.

Moss, S. (2008): "Alternative approaches to the empirical validation of agent-based models." *Journal of Artificial Societies and Social Simulation* **11(1)**: p. 5.

Mueller, M. G. & P. de Haan (2009): "How much do incentives affect car purchase? agent-based microsimulation of consumer choice of new cars—part

i: Model structure, simulation of bounded rationality, and model validation." *Energy Policy* **37(3)**: pp. 1072–1082.

Pearson, C. (2014): "Engine wars! cryengine shifting to cheap subscription." `http://http://www.rockpapershotgun.com/2014/03/20/cryengine-subscription-model/`. Accessed: 2015-05-07.

Said, L. B., A. Drogoul, & T. Bouron (2001): "Multi-agent based simulation of consumer behaviour: Towards a new marketing approach." In "International Congress on Modelling and Simulation Proceedings," .

Smith, A. (1776): *An Inquiry into the Nature and Causes of the Wealth of Nations*. T. Nelson and Sons.

Szyperski, C. (2002): *Component software: beyond object-oriented programming*. Pearson Education.

Tisue, S. & U. Wilensky (2004): "Netlogo: A simple environment for modeling complexity." In "International conference on complex systems," pp. 16–21.

Windrum, P., G. Fagiolo, & A. Moneta (2007): "Empirical validation of agent-based models: Alternatives and prospects." *Journal of Artificial Societies and Social Simulation* **10(2)**: p. 8.

Zeeman, E. C. (1974): "On the unstable behaviour of stock exchanges." *Journal of mathematical economics* **1(1)**: pp. 39–49.

Zhang, T. & D. Zhang (2007): "Agent-based simulation of consumer purchase decision-making and the decoy effect." *Journal of Business Research* **60(8)**: pp. 912–922.

Appendix A

Important Code

In this appendix the implementation of the methods concerning the decision of the customer can be found. The complete source code is too large to be listed here.

Listing A.1: Code of function MakeChoice

```
public Product MakeChoice(ObservableCollection<Product> offersList)
{
        foreach (Product product in offersList)
        {
                if (!Perceptions.ContainsKey(product))
                {
                        //create new ProductPerception
                        ProductPerception p = new ProductPerception();
                        Perceptions.Add(product, p);
                }

                ProductPerception perception = Perceptions[product];

                ComputePerception(product, perception);

                ComputeMotivation(product, perception);

        }
        Product oldChosenProduct = this.ChosenProduct;
        if (oldChosenProduct != null)
        {
                int noChangeProb = (int)
                ((Math.Atan(Perceptions[oldChosenProduct]
                .Satisfaction/200) + Math.PI/2)/3.5*100);
                int random = rnd.Next(101);
                if (random > noChangeProb)
                this.ChosenProduct = decision.ChooseProduct(Perceptions);
        }
        else
        {
                this.ChosenProduct = decision.ChooseProduct(Perceptions);
        }
```

```
        if (this.ChosenProduct != oldChosenProduct)
        {
                this.ChosenProduct.NumberOfUsers++;
                this.ChosenProduct.LastIncome += this.ChosenProduct.Price;
                this.ChosenProduct.TotalIncome += this.ChosenProduct.Price;
                this.ChosenProduct.TotalSold += 1;
                if (oldChosenProduct != null)
                oldChosenProduct.NumberOfUsers--;
        }

        ComputeSatisfaction();
        return this.ChosenProduct;

}
```

Listing A.2: Code of function for computing Motivation

```
    private int ComputeMotivation(Product product, ProductPerception
        perception)
    {

        perception.Motivation += PriceSensitivity * product.Price +
            perception.Perception;

        return perception.Motivation;

    }
```

Listing A.3: Code of function for computing Perception

```
    private int ComputePerception(Product product, ProductPerception
        perception)
    {
        int perceptionImmediate;

        int qualityPerception = 0;
        int friendsPerception = 0;

        int change;

        change = (product.QualityParameterA - QualityRequirementA);
        if (change < 0)
        change = -change * change;

        qualityPerception += change * QualitySensitivity;

        change = (product.QualityParameterB - QualityRequirementB);
        if (change < 0)
        change = -change * change;

        qualityPerception += change * QualitySensitivity;

        change = (product.QualityParameterC - QualityRequirementC);
        if (change < 0)
```

```
    change = -change * change;

    qualityPerception += change * QualitySensitivity;


    foreach (Customer c in this.Friends)
    {
            if (c.ChosenProduct == product)
            friendsPerception +=
                ((c.Perceptions[product].Perception*FollowerTendency)
                /100)/this.Friends.Count;
    }

    perceptionImmediate = MarketingSensitivity * (friendsPerception +
        qualityPerception) / 100;

    perception.Perception = (Memory * perception.Perception) / 100 +
        perceptionImmediate + perception.Satisfaction;

    return perception.Perception;
}
```

Listing A.4: Code of function for computing Satisfaction

```
private void ComputeSatisfaction()
{
    ProductPerception perception = Perceptions[this.ChosenProduct];

    int satisfactionChange = 0;
    int change;

    change = (ChosenProduct.QualityParameterA - QualityRequirementA);
    if (change < 0)
    change = - change*change;

    satisfactionChange += change*QualitySensitivity;

    change = (ChosenProduct.QualityParameterB - QualityRequirementB);
    if (change < 0)
    change = - change*change;

    satisfactionChange += change*QualitySensitivity;

    change = (ChosenProduct.QualityParameterC - QualityRequirementC);
    if (change < 0)
    change = - change*change;

    satisfactionChange += change*QualitySensitivity;


    perception.Satisfaction =  (Memory*perception.Satisfaction)/100 +
        satisfactionChange;
}
```

Appendix B

Content of Enclosed Zip Archive

There is a zip archive delivered together with this thesis which contains the application, complete source codes and data from the simulations in this thesis. Our application can also be downloaded from the link provided in Chapter 5. The enclosed zip archive contains the following folders:

- Source: Complete source code of the application.
- App: Application with an executable file to run it.
- Data: CSV data from the simulations.