Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

COMPLEX SYSTEM SIMULATION: AGENT-BASED MODELING AND SYSTEM DYNAMICS.

Luisanna Cocco

Advisor: Giulio Concas
Co-Advisor: Michele Marchesi
Curriculum: ING-INF/05 Sistemi di Elaborazione delle Informazioni.

XXV Cycle
April 2013

Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

COMPLEX SYSTEM SIMULATION: AGENT-BASED MODELING AND SYSTEM DYNAMICS.

Luisanna Cocco

Advisor: Giulio Concas
Co-Advisor: Michele Marchesi
Curriculum: ING-INF/05 Sistemi di Elaborazione delle Informazioni.

XXV Cycle
April 2013

Dedicated to my family

Contents

1 INTRODUCTION. **1**
- 1.1 Complex systems and simulation. . . . 1
- 1.2 The most common simulation techniques. . . . 2
- 1.3 Outline. . . . 4

I BUSINESS MODELS. **7**

2 Introduction to the Business Models Proposed. **9**
- 2.1 A brief overview about the business models proposed. . . . 9
- 2.2 Background. . . . 12
  - 2.2.1 A CRM market analysis. . . . 12
  - 2.2.2 A literature analysis. . . . 19

3 OP and OD CRM Software Market Simulation Model. **29**
- 3.1 The Model proposed. . . . 30
  - 3.1.1 On-Premise and On-Demand vendors. . . . 31
  - 3.1.2 Investment policies. . . . 33
  - 3.1.3 Pricing policies. . . . 35
  - 3.1.4 Customers. . . . 38
- 3.2 A simulation-based approach to solve the proposed model. . . . 40
  - 3.2.1 Base Run results. . . . 41
  - 3.2.2 Sensitivity analysis. . . . 46
  - 3.2.3 Monte Carlo Analysis. . . . 52
- 3.3 Summary. . . . 56

4 OD and OD FLOSS CRM Software Market Simulation Model **59**
- 4.1 The model proposed: firms and customers. . . . 61
- 4.2 Base Run and other simulation set results. . . . 61
  - 4.2.1 Sensitivity Analysis. . . . 65
  - 4.2.2 Monte Carlo Analysis 70
- 4.3 Summary. . . . 71

5 Comparison among the Results the Simulation Models. **73**
- 5.1 Some considerations about the results in chapters 3 and 4 73

II SOFTWARE DEVELOPMENT SIMULATION MODELS **75**

6 Introduction to the Software Development Simulation Models Proposed. **77**
6.1 Background. 77

7 Kanban and Scrum vs Waterfall **79**
7.1 An overview of Waterfall, Lean-Kanban and Scrum processes 79
7.2 Model structure 81
7.3 Results 84
7.4 Summary. 86

8 Global Software Development with CC **91**
8.1 An overview of Global Software Development. 92
8.2 Model structure 94
8.3 Results 97
8.4 Summary. 99

III CONCLUSIONS. **101**

9 Concluding remarks **103**

Bibliography **107**

A Extra Data **117**
A.1 Chapter 3 Extra Data 117
A.2 Chapter 4 Extra Data 123

List of Figures

3.1 Quality of the OP primary (dashed lines), OP secondary (dotted lines) and SaaS (solid lines) products (***Base Run***). 44
3.2 Price of the OP primary and secondary products (***Base Run***). 44
3.3 Price of SaaS products (***Base Run***). 45
3.4 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (***Base Run***). The figure below expands y axis to highlight the customer distributions with smaller values. 46
3.5 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 4*). . . . 48
3.6 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 3*). The figure below expands y axis to highlight the customer distributions with smaller values. 49
3.7 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 5*). The figure below expands y axis to highlight the customer distributions with smaller values. 50
3.8 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 5.1*). The figure below expands y axis to highlight the customer distributions with smaller values. 51
3.9 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 5.2*). The figure below expands y axis to highlight the customer distributions with smaller values. 52
3.10 Customer distributions among the OP (dotted lines) and SaaS (solid lines) firms (*Set 6*). . 53
3.11 Customer distributions among the OP (dotted lines) and SaaS (solid lines) firms (*Set 6.1*). 53
3.12 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 7.1.1*). The figure below expands y axis to highlight the customer distributions with smaller values. 54
3.13 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 7.2.1*). The figure below expand y axis to highlight the customer distributions with smaller values. 54

4.1 Quality of CS (solid lines) and OS (dotted lines) product (***Base Run***). 64
4.2 Price of CS products (***Base Run***). 65
4.3 Price of OS products (***Base Run***). 66
4.4 Customers of CS (solid lines) and OS (dotted lines) firms (***Base Run***). 66
4.5 Clients of CS (solid lines) and OS (dotted lines) firms (*Set A*). 67
4.6 Clients of CS (solid lines) and OS (dotted lines) firms (*Set C*). 67

4.7 Clients of CS (solid lines) and OS (dotted lines) firms (*Set B.1*). 68
4.8 Clients of CS (solid lines) and OS (dotted lines) firms (*Set B.2.*) 68
4.9 Clients of CS (solid lines) and OS (dotted lines) firms (*Set D*). 69
4.10 Clients of CS (solid lines) and OS (dotted lines) firms (*Set E*). 69
4.11 Clients of CS (solid lines) and OS (dotted lines) firms (*Set F*). 70

7.1 Simplified version of the structure of the proposed model. 83
7.2 Causal loop diagram of error rate. 85
7.3 Delays in the proposed System Dynamic model. 86
7.4 *Rework in Scrum approach* 86
7.5 *Rework in Waterfall* 87
7.6 Selected Requirements in the three approaches. 87
7.7 Selected Requirements in Scrum and Lean-Kanban. 88
7.8 Work Done. 88
7.9 Live. 89

8.1 Simplified version of the proposed model. 94
8.2 Delays in the planning. 95
8.3 Development phase. 96
8.4 Delay for meetings. 97
8.5 Causal loop diagram of error rate. 97
8.6 Delay for testing and production environments. 98
8.7 Work done. 98
8.8 Live. 99

A.1 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 2*). The figure below expands y axis to highlight the customer distributions with smaller values. 117
A.2 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 2.1*). The figure below expands y axis to highlight the customer distributions with smaller values. 118
A.3 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 2.2*). The figure below expands y axis to highlight the customer distributions with smaller values. 118
A.4 Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (Set 5.3). The figure below expands y axis to highlight the customer distributions with smaller values. 119

List of Tables

3.1 *Parameters of the model related to the customers.* 42
3.2 *Values of the parameters of On-Premise and On-demand vendor equations.* 42
3.3 *Parameters common to both types of vendors.* 43
3.4 *Number of customers in the different firms in the market for the* ***Base Run****.* 45
3.5 *Number of customers in the firms in the market for sets 2, and 2.1.* 47
3.6 *Number of customers in the different firms in the market for sets 2.2, and 2.3.* . . . 48
3.7 *Number of customers of the two type of firms for sets 2.i, 3, 4, 5.i, 6, 7, and Set 8.* . . 49
3.8 *Number of customers in the different firms in the market for the Sets 7.i.* 55
3.9 *Number of customers of the two type of firms in the market for Set 7.i, and Set 8.i.* . . 55
3.10 *Sets of Simulation.* 56
3.11 *Monte Carlo analysis: Percentiles of the number of customers and total average number of survivor firms for Base Run', and Base Run'.1.* 56
3.12 *Monte Carlo analysis: Percentiles of the number of customers and total average number of survivor firms for Base Run', and Base Run'.1.* 57

4.1 *Pricing Options for CRM products [64].* 60
4.2 *The table reports the values of the parameters of On-demand vendor equations.* . . 63
4.3 *The table reports the values of the parameters of On-demand vendor equations.* . . 64
4.4 *Quality Values of best OS and CS products in the market.* 65
4.5 *Monte Carlo analysis: Percentiles of the number of customers and total average number of firms survived for sets B, and D.* 70
4.6 *Monte Carlo analysis: Percentiles of the number of customers and total average number of firms survived for sets E, and F.* 71

7.1 *Initial Parameters of the proposed model.* 84

A.1 *Number of customers in the different firms for sets 5, 5.1, and 5.2, and instants t_{entry} and t_{exit} for the firms ousted from the market.* 119
A.2 *Number of customers in the different firms for Sets 6.i, and instants t_{entry} and t_{exit} for the firms ousted from the market.* 120
A.3 *Number of customers in the different firms for sets 7, 7.1, and 7.2, and instants t_{entry} and t_{exit} for the firms ousted from the market.* 120
A.4 *Number of customers in the different firms for sets 7.1.1, and 7.2.1, and instants t_{entry} and t_{exit} for the firms ousted from the market.* 121
A.5 *Number of customers in the different firms for sets 8, 8.1, and 8.2, and instants t_{entry} and t_{exit} for the firms ousted from the market.* 122

A.6 *Number of customers of the different firms for sets A, B and C* 123
A.7 *Number of customers of the different firms in the market for sets B.1 and B.2.* 123
A.8 *Number of customers of the different firms in the market for sets D, E and F* 124

Chapter 1

INTRODUCTION.

1.1 Complex systems and simulation.

Complex and long systems of nonlinear equations will be an image of little significant if we do not know that they hide realities, just waiting to be uncovered. Modeling a complex system with a large system of nonlinear equations may in fact hide the inner working of the system behavior, even if these equations perform well in modeling it. The relationships among inner components would remain unknown if the system is just described through closed form analysis. Moreover, many systems cannot be effectively described by closed mathematical models, and the models describing other systems nor have any analytical solution, neither have a simple and practicable numerical solution. This happens with many nonlinear differential equations, and with stochastic models. This fact leads us to adopt a simulation approach for studying these complex systems.
Simulation is the set of processes which aim to design and implement a model of a real system. The purpose is to understand the functioning of the system or to evaluate different strategies to define the best effectiveness of the system itself.
Simulation through Computational Sciences represents a new way of doing science. It is a discipline which has its roots in and is based on mathematics and computer science. It tries to translate the variety and the complexity of the world in arrays of numbers, algorithms, and models which are understandable by a computer. The main advantage of such simulation is that the experiments can be fully controlled and all the performance of the system observed. Moreover, it is also the best tool for analyzing dynamic, time-varying or stochastic systems, in which changes can occur in unpredictable way.

In recent years particular attention was devoted to the simulation-based approach in the economic field. According to complexity theory, the socio-economic system is a complex adaptive system, consisting of a large number of individuals who interacting among themselves create unpredictable behaviour. With the heterogeneous-agent-based models the diversity in the various market participants or in the system under test is explicitly taken into account. These models are not linear and this implies that neither their outcomes, nor their operation can be easily modeled and reproduced in a set of equations. They allow us to obtain information on the behaviour and status of the operators and can explain the interactions of the market and provide better forecasting tools. One of the drawbacks lies in having to make long and heavy development work, although the key to success is inherent in as realistic a description as possible, which makes few assumptions. While making such a description we should try not to reproduce too specific cases, which burden and complicate the model. Of course, the simulation also has some limitations.

- It is neither a science nor an art, but, unfortunately, a combination of both; experience is fundamental.
- It is an iterative and experimental technique.
- It is expensive in terms of development of simulators and processing time.
- It is difficult to convince others of its convenience.
- The validation of the data reliability is particularly critical.
- The analysis and the interpretation of the results requires a good knowledge of statistics.
- The results can be misinterpreted and it is difficult to identify the sources of error.

1.2 The most common simulation techniques.

The simulation is a methodology of experimental analysis of dynamical systems and in particular of complex dynamic systems [49]. A complex dynamic system is a set of entity which although are distinct, interact through relationships of interdependence or reciprocal connection. For dynamic systems we refer to systems characterized by evolution over time. The focus is not placed on the analysis of the system in its equilibrium state, but on the path through which it changes over time. The network of relationships between the entities produces non-linear effects, which can not be explained by studying each component individually.

The presence of nonlinearity, the dominant feature in complex systems, leads us to adopt the simulation as an interesting alternative to analytical models in the study of complex dynamic systems. With a simulation model we are able to represent the system in a brief way, and to study the aggregate behaviour.

The simulation is a methodology which is part of the so-called experimental mathematics. It represents a system through a computer system, using the computer for computing numerically the behaviour of the system. Through programming languages we can define the properties of a system in detail, determining the behaviour dynamically, on the basis of its current state. The analytical methods often require the adoption of generalized descriptions of the behaviour of the model, or the building of complicated systems of differential equations, capable of describing different behaviours for different intervals of the state variables. Instead, in the simulation approach through conditional constructs, typical of programming languages, we can introduce with simplicity such conditional behaviours.

In spite of these advantages, the simulation involves difficulty in the generalization of the results. Pulling out the laws of a general simulation model is certainly more difficult than pulling them out from an analytical model. In fact, if we find the solution of the system of equations, we have full information on the system represented. The simulation, instead, is only able to provide information on individual instances of a possible future path of the model, often determined by the initial conditions. Therefore, this methodology does not

provide the same quality and information of an analytical solution, but certainly it allows us to analyze and formalize complex systems, which are otherwise intractable.
Now, we present two of the most wide numerical simulation techniques based on computer science:

- ***agent based modeling***, and
- ***system dynamics***.

Agent-Based Modeling (ABM) is a methodology for representing simulation models based on object-oriented programming. Until the 90s, the dominant programming technique was the imperative programming based on the concept of function. Functional programming can produce an ordered code and allows us to define libraries of reusable functions.
In the early 70s a revolutionary programming paradigm was developed: OOP (object-oriented programming). Object-oriented programming introduced the concept of class as an evolution of the function.

The class defines a conceptual category, from which one or more items can be instantiated. This paradigm gave a great impetus to the development of complex applications, and also was the source of inspiration for researchers at the Santa Fe Institute (Institute for the Study of Complexity), who created a metaphor between software objects and agents of social systems.

The code representing a class of objects is unique, but through it, we can create an infinite number of independent objects, each characterized by its different experience. Creating n agents from one or more classes we obtain multi-agent systems, where individual entities are different from each other and evolve in different ways depending on their initial parameters and on the interactions which happen during the evolution of the simulation.

The agent-based models are mostly continuous models. At regular intervals, the system repeats one or more operations with a precise sequence. However, given the flexibility of the ABM technique, the possibility to create models with discrete events is not excluded. In agent models we can distinguish four categories of classes.

- Classes of service are used by the simulation program to perform technical activities not strictly related to the logic of the system simulated. For example, classes used for accounting, to define spatial structures used to manage statistics, and more.
- Agent is a class which corresponds to an homologous entity which we can observe in the real system.
- The model is a special class which coordinates the simulation. It deals with the creation of the objects from classes, sets the simulation parameters and defines the structure of the events which determine the dynamics of the model.
- The observer is a class which does not interfere with the model but it takes care of reading the status of the variables and of plotting the simulation to the user.

Although the agent-based models are used to represent any system, they are particularly suited to representing social systems, or environments where the entities are very heterogeneous, and complex phenomena occurr.

System Dynamics introduced in 1961 from Jay Forrester at MIT (Massachusetts Institute of

Technology) derive from the study of control systems. Among the techniques of simulation is the one that more similar to mathematical formalism, since it is based on differential equations. It provides a reduced use of programming languages, enabling great celerity in the design of the models. A differential model is based on equations in continuos time, whose computer simulation always involves a discrete treatment. In fact, any technique of computer simulation must coexist with two physical limits:

- the discreet representation of numerical quantities and therefore also of the time; and
- the lack of parallelism in the performance: the events which in reality we perceive as simultaneous can not be represented so in the computer. In fact, the simulator always makes a choice on the sequence of actions to be taken.

In the formalism of system dynamics there are three types of variables: stock, flows and auxiliary variables. The level variables or stock refer to the stock of a good at a given time t, acting as containers, which are filled and emptied during the evolution of the system. The flow variables represent the rate at which a variable level changes over time. The net rate of change of a stock is the sum of all the flows in input less the sum of all the flows in output. The rate represented by the flow variables can be expressed as a constant value, a stochastic function or may depend on other variables in the model.

The system dynamics can represent positive and negative feedback. The feedback loop highlights the presence of relations which self-feed positive feedback or negative feedback.

1.3 Outline.

In this thesis, the simulation based approach is applied to the study of the software market and the methodologies for software development, allowing us to highlight results which would otherwise remain obscure. This thesis is divided in two parts.
In the ***first part*** a model to study and simulate the software market is presented. The model has been developed through several steps: starting from quite a simple model, implemented in Matlab code, we arrived at a much more complex and realistic model implemented in Java code. The several steps have been published in some scientific publications (see [35], [36]) which gave value to the work and paid off for sustained commitment. In the first part of the thesis, we present the final and complete version of the model, applied to two different study cases. The first part of this thesis is organized as follows.
In Chapter 2, we give brief overview about the business models proposed and the background to the business models.
In Chapter 3, we propose a model to study the competition among On-Premise (OP) and On-Demand(OD) vendors who offered CRM software solutions.
In Chapter 4, we propose a modified version of the model presented in the previous chapter for analyzing and studying the competition among On-Demand firms offering Customer Relationship Management (CRM) products with or without the source code availability.
Finally, in chapter 5, we do a brief comparison between the results obtained simulating the two previous models.
Our *goal* is to propose a model to analyse and study the software market, in particular the CRM software model. A model which could be used to forecast the market trends, to plan the investment and pricing business policies and business winning strategies.

In the ***second part***, the simulation of some methodologies for the software development is dealt with. In particular, we conducted an analysis on Agile methodologies versus traditional methodologies, and an analysis on Cloud Computing applied to Global Software Development. More attention was given to the first part than second, therefore this last requires further studies and future research. Also these topics have been the subject of some scientific publications (see [37] and [38]). The second part is organized as follows.
In Chapter 6, we give brief overview about the software development simulation models proposed and the background to the models.
In Chapter 7, we analyze the dynamic behaviour of the adoption of Kanban and Scrum, versus a traditional software development process such as the Waterfall approach. We use a system dynamics model, based on the relationships between system variables, to assess the relative benefits of the studied approaches. The model is simulated using a commercial tool, Vensim. The proposed model visualizes the relationships among these software development processes, and can be used to study their relative advantages and disadvantages.
In Chapter 8, we propose a model for studying how Global Software Development can be facilitated using Cloud development environments, compared to a more traditional development environment. We use System Dynamics to model and simulate how effective Cloud-based software development environments are for Global Software Development. Both studied environments assume a development process based on Scrum agile methodology. The proposed model could be used as a tool by the project managers to understand how Cloud Development Environments might facilitate Global Software Development.
The *goal* of the first model is to propose a software development model which allows us to highlight the strengths and weaknesses of the software development different methodologies.
The *goal* of the second model is to highlight how a Global Software Development environment on the Cloud Platform may facilitate Global Software Development with respect to an environment set up on-premise. The simple tools proposed can be easily customized and used by every small or medium enterprises to monitor their business processes of software development.
Finally, the thesis concludes with the Chapter 9, where we report some final considerations about all the topics dealt with.

Part I

BUSINESS MODELS.

Chapter 2

Introduction to the Business Models Proposed.

2.1 A brief overview about the business models proposed.

In this thesis, we propose a model to represent a vertical software market, that is a segment of the software market in which some applications compete, giving functionalities to perform a specific job. These are applications specifically dedicated to a specific manufacturing or services sector (for example, footwear, pharmaceuticals, food, metalworking,or credit), and to specific companies, usually medium or small companies. Examples of vertical applications are those which allow a doctor to manage medical records, and an insurer to manage the issuance of insurance policies. Taking the first example into consideration, the software is specifically designed to be used in a clinic, hardly it will be useful for different realities. The software can be used so as it has been released, or a doctor can hire a consultant to modify it to meet its specific needs. This second possibility is particularly easy in the case in which the application is FLOSS.
Customer Relationship Management software, CRM, is a good example of a vertical software solution.
CRM is a successful business strategy, a new method of work and processes management, which through the achievement of organizational efficiency, allows us to increase the company's turnover ensuring a high level of customer satisfaction. The CRM provides a new approach to the market, which puts the customer and not the product at the center of the business.
In recent years CRM market has continued to prove its worth with some impressive growth. According to Gartner's 2012 Chief Executive Officer (CEO) Survey, CEOs valuated enterprise suite CRM as the most important area of investment to improve their business over the next five years.
We can distinguish four main categories of CRM [21]: Enterprise Suite CRM, Midmarket Suite CRM, Small-Business Suite CRM and Open Source CRM. Moreover, three are the ways to distribute the CRM solutions: SaaS, On-Premise and Open Source.

- **CRM SaaS**, sometimes called On-Demand, hosted delivery or cloud computing, permits businesses to rent access to CRM software systems and the IT infrastructure which delivers those systems. The customers pay for the business systems with a per user subscription fee and access to information from any location at any time;

- **On-Premise CRM solutions** are formed by closed proprietary systems. They are expensive to purchase, customize and administer. They are difficult to install, and are released under the payment of a software licence fee;
- **Open-source CRM solutions** are often far less expensive and easier to modify than their proprietary counterparts, and if some modifications are needed after deployment, programmers are usually easy to find.

In 2012 the leaders of the Enterprise Suite CRM market were Microsoft, NetSuite, Oracle and SAP [26]. The leaders for Midmarket Suite CRM were Microsoft, NetSuite, Oracle and SugarCRM; those for Small Businesses Suite CRM were Microsoft, NetSuite, Salesforce and SugarCRM. Finally, those for Open Source CRM were Adempiere, Concursive, Consona and vTiger.
From [25] and [27], we know the market share for SAP, Oracle, Salesforce, Microsoft, Amdocs, and for the remaining smaller companies, and know that the in 2008 Salesforce.com acquires 51,800 customers launching CRM for Google Apps.
So we can estimate the number of thousand CRM customers for each firm: 110 for SAP; 79 for Oracle; 31 for Microsoft; 24 for Amdocs; 194 for others, the total number of customers in the CRM market being 490.

In this thesis, we present the modeling of the CRM software market, in which compete On-Demand and On-Premise CRM vendors.
The main agents modelled are software houses and enterprise customers. We represent the firms, their creation, their life and all their business activities. We modelled three type of vendors: On-Premise vendors, On-Demand CRM vendors offering their software solution with code availability, and On-Demand CRM vendors offering their software solution without code availability.
We start highlighting their main characteristics according to the papers [1], [56], [43]. The first aspect to highlight is surely the way in which the two types of firms deliver their products [62]. On-Premise products are on site applications, instead On-Demand products are applications delivered from multi-tenant system via the Internet. On-Premise vendors are only accountable for building and testing the software and for on-going support and bug-fixing. On the contrary, On-Demand vendors, called also SaaS providers, sell a complete service which includes delivery, support and on-going maintenance. Hence, the costs of the service delivering (costs of hosting, bringing customers, users onboard, cost of managing the application and data center environments), the costs of the on-going maintenance, and finally, the costs of the updating play a key role in the success of these vendors. These costs do not weight on the customer, as in the traditional models, but they weight directly on SaaS providers.

SaaS providers must assure the integration to application level, to client level and to solution level, and moreover, they must guarantee customizable solutions, as traditional On-Premise applications do. All this must be done without neglecting user satisfaction according to their service level agreements regarding uptime and performance.

Compared to perpetual license vendors, who collects high up-front revenues, SaaS vendors collect low revenues during the course of the subscription. For this reason, having ef-

ficient finance and accounting teams, which provide stable financial processes, is very important in this new pricing model.
One important advantage of SaaS vendors, with respect to traditional vendors, is to enjoy a lower ”time to market”. SaaS companies adopt an incremental R&D approach, which must match their revenue flow. Over time, the R&D will become a continuous-improvement approach. Time to market is low because the deployment cycles are short, and the new functionalities are marketed and delivered online.
Another important advantage is that the capital invested for R&D is small. Indeed, the market response to the different products can be tested, and only after the decision of investing large amount of capital can be taken into account. On the contrary, On-Premise vendors invest large quantities of capital for R&D, and develop and maintain multiple versions of a product to run on different platforms. SaaS vendors are divided between vendors who sell their product without the code availability, and vendors who distribute their product with code availability. In the last case, the user is relatively independent from the provider. Consequently, if user wishes to fix a bug immediatly, he can make the fix by himself, he can obtain the fix from the open source development community, or he can wait for the fix to be included in a future release. These software solutions offer a lower annual cost than the first solution, and in addition, they offer several benefits associated to the use of open source software.
Generally speaking, a business can be defined as an organized center for the systematic production of goods and services, which aims through the exchange, to achieve a surplus, which is the excess value produced with respect to the values consumed to produce.
The birth of a company is closely related to a feasibility analysis. The feasibility analysis and the development of the activity proceed according to three guidelines, which cover three fundamental aspects of every business activity: technical, economic and financial aspects. Each guideline is well defined in a specific document: the project, the business plan and the financial plan, respectively. The project summarizes all the key solutions for the subsequent economic evaluations. The business plan outlines the costs and revenues which will be produced by the investment; finally, the financing plan examines the cash flows and, therefore, the return on investment.

Today, companies are facing a competitive market like never before. The development of the Internet, changing demographics, increased competition and over productive capacity are just some of the factors affecting organizations’ business. The main goal of a company is not just finding new customers but to retain the loyalty of its most profitable customers. Each business unit can be successful only if each one is in syncrony with the others, and competitive advantages over competitors can be achieved by meeting promptly the customers’ needs. To be successful in the current competitive market, companies must be able not only to produce goods and/or services, but also to increase the customer value and his degree of satisfaction. The most recent research shows that the winning choice is of creating, developing and maintaining deep relationships with established and potential customers.

A business process is a set of interrelated activities carried out within the company, which create value by transforming resources into a product for customer. Hence, all the investments done by a firm aim to create competitive and customized products.
Surely, modeling a software house is not easy; there are many aspects to consider and many of them are very difficult to model. For this reason, implementing all the aspects characterizing a software vendor is impossible.
In the next Chapter, we present in detail how we modelled a software vendor, its activities

and its interactions with customers. Many are the semplifications done, and so the products, the investment and pricing policies, and purchase choices of the customers modelled aim to simulate the behaviour of the different agents in as realistic a way as possible.
We modelled the CRM market using the heterogeneous - agent - based models.
Our goal is to propose a model to analyse and study the software market, proposing it also as a tool to forecast the market trends, the investment and pricing policies and business winning strategies.

Surely, it should be obvious that more insights in the underlying processes are needed to give more precise and more conclusive results on the model. This, however, also requires more and better data.

2.2 Background.

The structure of the business model presented in this thesis is built on many other papers written on the subject. Further, our model has been enriched and made as realistic as possible thanks to information and data coming from the market analysis.

Starting from the many insights from literature, and market analysis, we draw a model to study and analyze the competition between different types of vendors. Beginning from a simple model, which studies and analyses the competition between On-Premise and Floss vendors [35], we went along for steps, improving the model at each step, to simulate a more and more realistic software market.
We realized other three works. In the first, "Agent-Based Modelling and Simulation of the Software Market, Including Open Source Vendors" [36], we studied the competition between On-Premise and Floss vendors again, using a more realistic simulator than the first; in the second, "Simulation of the Competition among Traditional and On-Demand Software Vendors " [39], we defined a model for studying and analysing the competition between On-Premise and On-Demand CRM vendors. Finally, in the third, "Simulation of the On Demand Software Market, Including Vendors Offering Source Code Availability" [40], we studied and analysed the competition between On-Demand CRM vendors who offer their software solution with code availability, and On-Demand CRM vendors who offer their software solution without code availability. In this thesis, we described only the last two works in detail.

2.2.1 A CRM market analysis.

Before illustrating some of the leaders in the CRM, we briefly describe the concept of Customer relationship management software solution, which is related to the concept of customer loyalty.

In a market-oriented company the market is no longer represented only by the customer but also by the surrounding environment, with which the company must establish lasting relationships, taking into account the values of the client and society. So the focus toward the customer is crucial and decisive. For this reason, marketing management must plan and implement appropriate strategies to manage a resource so important.

The CRM goes substantially along four different and separated directions:

- the acquisition of new customers;
- the increase in the relations with the more important customer;

- the longest possible retention of customers who have more relationship with the company;
- the transformation of existing customers in people who praise the company and suggest their own company to people.

There are three types of CRM:

- Operational CRM: methodological and technological solutions to automate business processes, which involve the direct contact with the customer.
- Analytic CRM: procedures and tools to improve customer knowledge through the extraction of data from operational CRM, their analysis and the prognostic study on the behaviour of customers.
- Collaborative CRM: methods and technologies integrated with communication tools (phone, fax, e-mail, etc..) to manage the customer contact.

A good CRM system includes a number of facilities both in the front office (in relation with the outside itself), and in the back office to analyze and measure data and the results achieved. There are many tools available to individual businesses in order to establish an individual relationship with the customer, for example:

- chat online;
- discussion;
- a database containing the answers to the most frequently questions asked by users (FAQ);
- an e-mail to be addressed;
- information services provided on other means (such as SMS to send to your mobile phone, or the use of technology WAP)
- Ticket on-line for reporting problems or assistance requesting;
- Internal tracking of every communication "from" and "to" the customer;
- Estimates and Invoices addressed to the customer;
- History of payments made by the customer;
- Analysis of navigation for profiled users with the aid of web analyzer;
- Social networks.

Now, we report some information about the CRM market drawn out from some web sites [21] and [25].

Gartner estimates 35% of all CRM software is now consumed using SaaS, and expects that figure to grow to just over 50% by 2020. The four fastest-growing SaaS sectors are sales, social CRM, customer service and marketing. In fact, marketing automation is the most rapidly growing, up from 19% using SaaS marketing software in 2010, to 29% in 2011 and to over one-third in 2012. Gartner, Forrester, IDC and AMI agree that the new sale applications will be procured via SaaS more so than on-premise sale applications.

Oracle, Salesforce.com, SAP and Microsoft are the 'Big 4' of the CRM software market, and the only enterprise software vendors to exceed $1 billion in CRM software revenues.

The top 10 CRM vendors made up about 49% of the CRM software market.

The competitive landscape for the worldwide CRM applications market is becoming decidedly more interesting. As regard the market share, the top vendors are inching closer together with just one or two percentage points separating the leaders. One area of differentiation among the leaders is the geography where market shares are built. While Oracle holds the lead in Asia/Pacific (excluding Japan), Salesforce.com is very strong in North America and Japan. SAP is the established leader in EMEA (Europe, Middle East, and Africa) and Latin America.

Ironically, while SaaS tends to make the more predictable CRM market in terms of revenue calculations and predictions, it also has the potential to disrupt the market share pecking order more quickly.
SaaS lowers the entry barriers to the customers, and unlike on-premise systems, which require months if not years to rip and replace projects with hefty capital expenditures, SaaS CRM systems can often be replaced in weeks and without significant capital expenditures or significant changes to operating expenses. This creates a new variable which likely will influence market share rankings and will alter the CRM leaderboard in the years to come.
In the next sections, we give an overview about the CRM firms which cover an important position in CRM market nowadays. We talk about SAP, Microsoft, Salesforce, NetSuite, SugarCRM, Adempiere, Concursive, vTiger, Consona and Zoho.

SAP.

SAP is a German multinational software corporation, which makes enterprise software to manage business operations and customer relations [28], [25]. Headquartered in Walldorf, Baden-Württemberg, with regional offices around the world, it is the largest software company in Europe and the fourth largest in the world. The company was founded in 1972, and operates three segments: Product, Consulting, and Training.

SAP currently has sales and development locations in more than 50 countries worldwide. SAP invests in 26 vertical markets in order to deliver industry specific Enterprise Resource Planning (ERP) solutions. The industries which seem to get the most attention include business services, consumer products, financial services (particularly banking), government/public sector, oil and gas, retail, telecommunications, transportation/logistics and utilities. The SAP partner channels and ecosystems provide wide support for additional industry solutions.

SAP counts over 102,000 customers of all sizes, industries and locations, and its software product portfolio includes:

- SAP Business Suite software for large organizations, designed for companies with more than 2,500 staff and selected vertical markets;

- SAP Business All-in-One solutions, on-premise ERP and CRM solutions, designed for upper middle market (from 500 to 2,500 staff);

- SAP Business ByDesign SaaS solution, software as a service (SaaS) ERP and CRM system, designed for lower middle market (from 100 to 500 staff); and
- SAP Business One application, on-premise ERP and CRM application, for SMB (small and midsize businesses, that is less than 100 staff).

Microsoft.

Microsoft was founded on 1975 headquartered in Redmond, WA. It is the largest software company in the world [25]. The company operates five business segments, including Windows & Windows Live Division, Server and Tools, Online Services Division, Microsoft Business Division, and Entertainment and Devices Division. Its CRM product is called *Dynamics CRM 2011.* It is designed for small and midsize business (SMB), and has positioned Microsoft to compete for top CRM software market position. Customer choice with delivery model- be it on-premise or on-demand- provide a unique competitive advantage.

Microsoft Dynamics CRM software capabilities include: marketing management, sales force automation (SFA), customer service, and system administration. Further, customers can subscribe to Dynamics CRM 2011 directly from Microsoft or from various Microsoft partners.

Salesforce

Salesforce was founded in March 1999 by former Oracle executive Marc Benioff along with Parker Harris, Dave Moellenhoff, and Frank Dominguez [25]. Salesforce.com is a provider of enterprise software applications delivered via the software-as-a-service or cloud computing model. The company's flagship product is a CRM system designed for businesses of all sizes and industries worldwide. The company also provides a platform-as-a-service (PaaS) solution titled Force.com and manages a portfolio of integrated third party applications referred to as AppExchange. Salesforce.com markets its CRM and enterprise software solutions to businesses on a subscription basis, primarily through the direct sales efforts and indirectly through a business partner channel.

Its products are:

- Sales Cloud which includes SFA, PRM, marketing, Chatter, Jigsaw, content library and mobile;
- Service Cloud SMB, a comprehensive call center case management solution with suite of online tools;
- Jigsaw Data Cloud, a mashup service to insert and append contacts as well as manage and clean contact database;
- Chatter Collaboration Cloud, a internal social communication tool to follow people, groups and individual CRM records;
- Force.com Custom Cloud SMB to Enterprise, a Platform-as-a-Service (PaaS) application development framework to develop or customize;
- Database.com a purpose-built cloud database to support online apps, social apps and mobility;

- RemedyForce a Help desk support application jointly developed by BMC and Salesforce.com; and finally
- AppExchange a third party directory and marketplace of online applications integrated with Salesforce.com.

NetSuite

NetSuite manufactures an on-demand and integrated business application suite which includes ERP, CRM and Ecommerce functionality to medium- sized businesses and divisions of large companies. The Company's products, including NetSuite, NetSuite OneWorld and NetSuite CRM+, are designed as an enterprise-wide business software suite delivered over the Internet as a subscription using the SaaS model [25]. The company was founded in 1998 by the current chairman and CTO, Evan Goldberg, with financing from Oracle founder and CEO Larry Ellison.

The company counts over 7,000 customers, most of them small and midsize businesses. The NetSuite ideal customer profile is the middle market company, or subsidiary of a large enterprise organization, who desires a single SaaS business application for both front office (CRM) and back office (ERP) operations.
Its products are:

- NetSuite CRM+ allows firms primary marketing automation and include campaign management, lead management&analytics;
- NetSuite includes CRM, Enterprise Resource Planning (ERP)/accounting software and e-commerce
- NetSuite OpenAir is a vertically focused Professional Service Automation (PSA) solution with tight integration to NetSuite;
- NetSuite OneWorld is a global business management and financial consolidation software system;
- NS-BOS Platform is a Platform as a Service (PaaS) development platform to extend, modify or adapt NetSuite;
- SuiteCloud is a developer network and program with tools and co-marketing for software developers;
- SuiteApp.com is an online marketplace of value-added integrated cloud solutions for unique business processes or industries;
- Unlimited Edition is a volume license that includes 23 modules and removes limits for users, subsidiaries and more.

SugarCRM

The *SugarCRM* [25] pursues maintaining viable company and delivering customer benefits attributable to open source software solutions.

SugarCRM has a two-tiered business model, the company gives away the basic open source customer relationship management software application, and sells four progressively richer commercial versions, which include proprietary code and more advanced feature sets such as integration to Microsoft Outlook, support for MS SQL or Oracle databases, disconnected mobile support, customer self service, reporting and system integration capabilities for companies wanting to achieve full CRM software potential.
The company was founded in April 2004 by John Roberts, Jacob Taylor and Clint Oram. Today this company retains over 7,000 customers from around the world. The SugarCRM ideal customer profile is the small to midsize business (SMB) seeking a flexible, open-source CRM solution available in a choice of deployment models (i.e. deploy on-premise, on-demand or a choice of public or private clouds).

Its products are:

- Sugar Community Edition which includes CRM software functions of sales force automation (SFA), marketing and service. It permits customization with Module Builder&Custom Fields;
- Sugar Professional Edition which expands upon Community Edition with new UI, deeper CRM feature sets, more integration, Mobile, workflow and more administrative management options;
- Sugar Corporate Edition which expands upon the Professional Edition with Mobile Plus, additional storage allocation and increased support availability and response SLA;
- Sugar Enterprise Edition which expands upon Corporate Edition with Enterprise Reporting, Customer Portal, offline client, Oracle or MS SQL support and additional customer support;
- Sugar Ultimate Edition which extends Enterprise Edition with Sugar Connector to Lotus Domino, additional storage allocation and increased support and response SLA.

Adempiere

The *Adempiere* project was created in September 2006 after a long running disagreement between ComPiere Inc., the developers of Compiere^{TM}, and the community which formed around that project [22]. The community believed Compiere Inc. placed too much emphasis on the open source lock-in/commercial nature of the project, rather than the community sharing/enriching nature of the project, and after an impassioned discussion decided to split from Compiere^{TM} giving birth to the ADempiere project. ADempiere Business Suite is an industrial strength open-source software solution which combines ERP, CRM and SCM support for business process. ADempiere provides a framework for extending and customizing to meet business needs.

Concursive

Concursive Corporation is a software and social media company which is part of Intel Capital's portfolio [23]. It is headquartered in Norfolk, Virginia and has offices in Melbourne, Australia and Bangalore, India. Concursive products are used by large enterprises, governments, universities and thousands of small businesses alike. It has more than 17,000 registered community members.

ConcourseSuite is a front office application suite to integrate CRM, web content management and team collaboration capabilities into a single, easy to user web application. An open source Java-based application with a standards-based plug-in architecture, ConcourseSuite can scale to the largest enterprise, yet delivers the ease of use and low cost required by small and medium-sized businesses.

ConcourseSuite comprises several fully integrated modules. CRM, Web and Team modules combine to meld traditional CRM functions with website authoring, content management, and Enterprise 2.0 collaboration features. With the Flex module, users can add custom functionality through the industry standard Portlet architecture.

vTiger

vTiger is an open source CRM software solution created from a fork of SugarCRM 1.0 [**?**]. Despite a mature product, global presence and use by over a 100,000 companies, often must to escape the shadows of competitor SugarCRM [25].
In the early years the company's revenue model was built around customer support and professional services such as implementation, integration and customization. In 2009 the company jumped into the SaaS market full steam and today the company is approaching 1,000 cloud CRM customers. The cloud CRM service is giving the company a new opportunity to scale.

The vtiger CRM solution is targeted to SMB's, who are often without internal IT resources. The company's largest customer concentration resides in the U.S., with about 40 percent of all customers located in America, about 30-40% of customers located in Europe and the remaining being widely distributed around the world.

The application includes the traditional CRM software tenants of sales, marketing and service, while also including less traditional functions such as quotes and sale order processing, inventory/products, billing and project management.

Consona.

Consona Corporation is a software company selling solutions to automate business critical tasks, ranging from marketing, service and support to planning and scheduling, material requirements planning (MRP), accounting, product configuration, and business intelligence [24].
The company both sells software and delivers some of its solutions as Software as a Service (SaaS) in a cloud environment. Its 4,500 customers operate in a variety of industries, from retail to automobile manufacturing, and range in size from small businesses to Global 2000 enterprises.
Consona employs more than 700 workers in 40 locations worldwide. The company maintains three of its own datacenters, two in the United States and one in Bangalore, India. Since 2001, Consona has grown tremendously through acquisitions. Its 2010 purchase of open-source and cloud ERP provider Compiere Inc., gave Consona its first cloud-ready ERP solution and entry into the distribution market.

Zoho.

Zoho is a software publisher of cloud based CRM software and an online office suite containing over 25 online applications including word processing, spreadsheets, presentations, databases, note-taking, wikis, project management, invoicing and other productivity applications [25].

ZOHO Corporation manages three divisions: WebNMS, which serves the needs of original equipment manufacturers (OEMs); ManageEngine, which delivers enterprise IT management; and Zoho.com, which delivers on-line business, productivity and collaboration applications.

Zoho headquarters are in Pleasanton, CA with additional offices in Austin, New Jersey, London, Tokyo and Beijing. Software development resides in Chennai, India. ZOHO is a tightly held privately owned company employing over 1,000 staff. The company claims it is profitable without ever having obtained outside capital.
The company was founded as AdventNet Inc. in 1996. In September 2005, Zoho launched its online applications business, and then changed its name to ZOHO Corporation in 2009 to better reflect the Zoho suite of on-line business applications.
Zoho states it has more than 50,000 customers. Customers are diversified, however, the majority are small businesses. Its products are:

- Zoho CRM Free Edition which includes CRM software functions of sales force automation (SFA), marketing and service, as well as integration with suite of online apps;
- Zoho CRM Professional Edition which includes CRM software suite of SFA, marketing and service, and inventory management, data management and enhanced user permissions;
- Zoho CRM which expands upon the Professional Edition with enhanced functionality user user management, data management and workflow automation.

2.2.2 A literature analysis.

The structure of the business model presented in this thesis is built on many other papers written on the subject.
Bonaccorsi et al. [4] proposed a simulation model in order to identify the relevant factors in the diffusion of Open Source, modelling the adoption decision of heterogeneous interacting agents.
Lihui Lin[34] studied how users' skill and network effects can influence the software market, characterized by proprietary firms, and open source firms.
Bitzer et al. [31] analyzed the influence of entry and competition of open source software on innovation and technological progress in software markets. They proposed a simple framework to examine a market structure where software producers compete in technology rather than in price or quantities.
Economides et al. [46] compared industry structures based on an open source platform with those based on a proprietary platform, analyzing the competition and the industry implication in terms of pricing, sales, profitability, and social welfare.
The work of Mustonen [44] explained the simultaneous existence of commercial alternatives to copylefted programs and why commercial alternatives to copyleft programs may not exist. In this model a monopolist firm invests in the quality of its program, that depends on the

programming output, and thus on the programmers' ability. The programmers can choose to develop for the monopolist firm or can be engaged in copyleft work, receiving complementary income based on their ability. Finally, the firm sets a price for its program, and the consumers value the various programs.
Almost all the works cited proposed mathematical models solved by analytical methods or simulation based methods, while we propose an heterogeneous agent-based model solved by a simulation-based approach.
From these works came the many insights which helped us in the fulfillment of our model. The works which suggested to us the basic ideas are the works by Haruvy et al. [16], and Gosh [52], which suggested some variables of our model, and the works by Bulcholtz [8], Yankee-Group [74] and Sugar provider [64], which suggested the orders of magnitude of the prices of the different software solutions.
In the following, we report a very detailed description of the first two works, while we briefly describe the other three works. In the first work, ***Open source development with a commercial product or service***, [16] the authors examined the optimal control decisions regarding pricing, the network size, and the hiring strategy, in the context of open source software development. This work examined a model in a monopoly setting, where the open source code is free, but complements another product which is sold comercially. The authors characterized price, quality, and hiring paths for firms under both the open source and closed source models. The optimal decision on opening the source depends on the importance of user contributions, wages and on the effectiveness of in-house developers.

When a firm decides to pursue a software project, it generally has an in-house developed prototype and some assessment of the in-house development potential.

If the potential contribution from outside programmers is not perceived to be large, obviously the firm should not pursue open source. Furthermore, when the benefit to the firm's complementary commercial product is small, the firm should keep the software product proprietary. The solution approach taken into account is the optimal control over a finite horizon. The firms vary the price of their commercial products over time, and plan in advance for the birth of new generations of products and the death of old ones.
Moreover, also the quality and the network size vary over time in the presence of network externalities. For network effect, all mean that the utility from a product increases with the number of other users. A network effect implies that the larger the network of existing users, the more likely non adopters are to adopt.
In the following, we describe how the authors modelled the choice between the open and closed source models.

The firm has two products, *product 1* and *product 2*. They have different codependent demand rates, D_1 and D_2, respectively. In the case of closed source development, all quality improvements arise in-house as a function of the number of in-house developers. In the case of open source development, all quality improvements come from the users, as a function of the size m of the network. The variables used by the authors are:

- the quality of the software at time t, $Q_1(t)$;
- the exogenously given quality of the complementary product at time t, $Q_2(t)$;
- the size of network of users (i.e., installed base) at time t, $m(t)$;
- the number of in-house developers at time t, $N(t)$;

- the price of the software at time t, $P_1(t)$;
- the price of the complementary product at time t, $P_2(t)$.

The profit in a given period is the revenue P_1D_1 from the software, plus the revenue P_2D_2 from the complementary product, minus development costs, if they are applicable, and minus adaptation cost of the software to the firm's other products. The adaptation cost is particularly relevant in the open source case, in which the source code is not necessarily developed with the firm's objectives in mind. In the open source case, P_1 is by definition zero, and so the revenue is only P_2D_2. This loss of revenue may be offset partly by the fact that development costs are zero. This is not to say that the firm does not incur additional cost in adapting the open source software to its products. The additional cost is modelled by $C_0 >= 0$, and is assumed to be a fixed cost.

In the closed source case, development costs increase with the number of in-house developers, N, and are assumed equal to a cost function wN^2, in accordance with the economic principle of increasing marginal cost, where w is not unit wage but rather a cost parameter, generalizing the cost function becomes wN^γ.

In the closed source case, the in-house contributors improve the quality of the software over time. That is,

$$\dot{Q}_1 = KN - \delta * Q_1 \tag{2.1}$$

where:

$$Q_1(0) = Q_1^0 >= 0 \tag{2.2}$$

The parameter $K > 0$ denotes the productivity or effectiveness of the in-house closed source programmers. When K is high, closed source programmers are very effective and open source development may not be warranted. Note that software quality depreciates over time at a constant proportional rate $0 < \delta < 1$. This means high quality becomes obsolete faster than low quality.

In the open source case, on the other hand, there are no in-house developers and all contribution is through users. Hence,

$$\dot{Q}_1 = \alpha m - \delta * Q_1 \tag{2.3}$$

where:

$$Q_1(0) = Q_1^0 >= 0 \tag{2.4}$$

The parameter $\alpha > 0$ represents the level of involvement by the open source user community, which includes users of both the software and of the complementary product. The size m of the network of users increases each period by the number of new users of both products and decreases by a percentage of the existing users discontinuing the use of the product. The users of both products need not be equally weighted. A user of the software may be more or less valuable to the network than a user of the complementary product. The parameter $a > 0$ measures this relative weight. Separability between demands for the two products in the network is assumed, though the two are complements. That is, a user who uses both products has the weight of $(1 + a)$ in the network. The parameter $0 < \epsilon < 1$ is the rate of depreciation of the network or the rate of exit. Hence,

$$\dot{m} = aD_1 + D_2 - \epsilon m \tag{2.5}$$

where:

$$m(0) = m^0 >= 0 \tag{2.6}$$

The demand of the complementary product is

$$D_2 = h(P_2, m, Q_2) \tag{2.7}$$

where:

$$h(P_2, m, Q_2) >= 0 \quad \left(\frac{\delta h}{\delta P_2}\right) <= 0 \quad \left(\frac{\delta h}{\delta m}\right) >= 0 \quad \left(\frac{\delta h}{\delta Q_2}\right) >= 0 \tag{2.8}$$

that is, the demand D_2 is a decreasing function of the price of the complementary product P_2, a increasing function of the size m of the network of users, and finally, a increasing of the quality Q_2 of the complementary product.

All this is due to the law of demand, which states that price and demand are inversely related, to consumers derive utilities from other consumers using the product through newsgroups, file sharing, increased service, and compatible goods, and finally to the assumption that people derive utility from quality.
The demand of the software is D_1

$$D_1 = D_2 g(P_1, Q_1) = h(P_2, m, Q_2) g(P_1, Q_1)$$

where

$$g(P_1, Q_1) >= 0, \left(\frac{\delta g}{\delta P_1}\right) <= 0 and \left(\frac{\delta g}{\delta Q_1}\right) >= 0$$

This means that D_1 is a decreasing function of the price of the software P_1 and a increasing function of the quality of the software Q_1.
In the open source case, as $P_1 = 0$, then $D_1 = D_2 g(0, Q_1)$. The multiplication of the function $g(P_1, Q_1)$ by D_2 represents the effect of complementarity. That is, the demand for the software product is a increasing function of the demand for the complementary product.

The two models proposed and resolved are:
the **open source model**:

$$\begin{array}{ll} Max \int_0^T e^{\rho t}(P_2 D_2) dt - C_0 + \sigma(Q_1(T), m(T)) e^{-\rho T} & \\ s.t. & \\ \dot{Q}_1 = \alpha m - \delta Q_1 & Q_1(0) = Q_1^0 >= 0 \\ \dot{m} = a D_1 + D_2 - \epsilon m & m(0) = m^0 >= 0 \\ P_2 >= 0 & \end{array} \tag{2.9}$$

where:

$$D_1 = D_2 g(0, Q_1) >= 0 \quad D_2 = h(P_2, m, Q_2) >= 0 \quad \left(\frac{\delta\sigma}{\delta Q_1}\right), \left(\frac{\delta\sigma}{\delta m}\right) >= 0 \tag{2.10}$$

and

$$\sigma(Q_1(T), m(T))$$

is the salvage value;
and the **closed source model**:

$$\begin{aligned} &Max\int_0^T e^{\rho t}(P_1D_1+P_2D_2-wN^2)dt+\sigma(Q_1(T),m(T))e^{-\rho T} \\ &\qquad\qquad s.t. \\ &\qquad \dot{Q}_1 = KN-\delta Q_1 \qquad\qquad Q_1(0)=Q_1^0>=0 \\ &\qquad \dot{m} = aD_1+D_2-\epsilon m \qquad\qquad m(0)=m^0>=0 \\ &\qquad P_1,P_2,N>=0 \end{aligned} \tag{2.11}$$

where:

$$D_1 = D_2 g(P_1,Q_1)>=0 \quad D_2=h(P_2,m,Q_2)>=0 \quad \left(\frac{\delta\sigma}{\delta Q_1}\right),\left(\frac{\delta\sigma}{\delta m}\right)>=0 \tag{2.12}$$

because product quality and the size of user network cannot in reality be negative, and subsidies are not allowed.
The two models above described are two problems of control, and were solved using the technique of Lagrange' multipliers. The optimal control framework proposed in this work is ideally suited to jointly examine optimal trajectories for price, quality and network size as well as for determining the marginal values of quality and network size.

The analytical derivation and simulations run demonstrated that there are scenarios where open source may be beneficial to a firm. On the other hand, the results demonstrate that under various conditions, open source may not be beneficial to a firm. In particular, open source community involvement is critical to the success of open source. Only above a critical level of community involvement does open source become a viable alternative to closed source.

In the second work, ***Economic impact of FLOSS on innovation and competitiveness of the EU ICT sector.*** **[52]**, the authours studied the role of Free Libre Open Source Software (FLOSS) in the economy, its direct impact on the ICT sector, and its indirect impact on the ICT-related sector. The model developed by UNU-MERIT refers to the economic theories of endogenous growth and contributions of Romer[2] [5].

According to Romer, the capital is not only physical / tangible capital (computers) but also intangible (knowledge). The production of the knowledge generates positive externalities. If a firm invests in R&D and creates a new product, even if it is recorded in the patent office, it does not prevent other companies to learn and imitate. Consequently, a company will use the knowledge produced by the other firms and will not pay anything. Thus, the production of each firm depends on physical capital, knowledge paid through the expenditure in R&D, and from work and from knowledge unpaid called also externalities.
The higher the investment in R&D made in the economy, the greater the opportunity for each individual entrepreneur to copy, emulate, inspire, generally to acquire knowledge not paid.
So, the economic development of a country depends on the investments done in R&D, on the externalities, but also on the growth rate of the per capita income which depends, always and positively, from the rate of savings.
In fact, the savings fund additional investments making it possible to increase production. Thus the endogenous growth models provide a theoretical justification for public intervention in support of growth.

In the model developed by UNU-MERIT [52] the labor represented from the human capital, is used to produce the final output, but also to perform R&D. This capital, considered as the general level of skills of an individual, is a factor of production which can be accumulated

as the physical capital, and acts on the productivity of a company. The increase of human capital stocks increases the R&S process, which becomes more productive, thereby increasing the technological capabilities of the economy and, consequently, also the production of the final output.
We briefly describe this model, starting from the definition of the effective capital stock:

$$K^e = (\int_0^{A^P} (x_(i)^P)^\beta di + \int_0^{A^F} (q * x_i^F)^\beta di)^{1/\beta} \tag{2.13}$$

where:

- A is the total number of different economic activities.
- A^P of these activities A are supported by PROPS and A^F by FLOSS, hence $A = A^P + A^F$.
- q is an index that represents the influence of quality differences in FLOSS and PROPS in turning physical capital into an effective input into the aggregator function.
- x_i^P is the amount of physical capital per PROPS supported economic activity,
- x_i^F is the amount of physical capital per FLOSS supported economic activity.

The companies use as input the work L and the capital k^e, hence the production function of Cobb-Douglas is given by:

$$Y = b * ((1-u) * h * L)^{1-\alpha} * (K^e)^\alpha \tag{2.14}$$

where:

- u is the fraction of time spent on human capital formation;
- L is the size of the population;
- h is the average human capital stock per person;
- α is a constant parameter in between zero and one;
- B is a positive scale parameter; and
- $((1-u)h * L)$ is the force used in the industry that produces the end good of the i_{th} in terms of human capital.

The production function must satisfy the macro-economic budget constraint:

$$Y = c * L + I + R \tag{2.15}$$

where:

- I represents current investment in physical capital;
- R represents current resources spent on aggregate ICT investment;
- c is consumption per head; and
- L is the number of the size of the labour force.

The growth of physical capital is equal to net investment, which in turn equals gross investment minus depreciation. So the physical capital stock grows in accordance with:

$$\left(\frac{dK_y}{dt}\right) = I - \delta^y \tag{2.16}$$

where:

- K_y represents the stock of physical (non-ICT) capital used in all final output producing activities taken together, and
- δ^y is the corresponding rate of depreciation of physical capital.

The aggregate ICT stock grows in accordance with:

$$\left(\frac{dK_i}{dt}\right) = R - \delta^i \tag{2.17}$$

where:

- K_i represents the stock of ICT capital, and
- δ^i is the corresponding rate of depreciation of ICT capital.

A fraction v of the stock of K_i is used in human capital formation, while the remainder is so divided: a fraction w is used in PROPS based final output production and a fraction $1-w$ in FLOSS based final output production. The investments in ICT capital, R, and investment in ICT capital, I, are fixed fractions s^R and s^I of gross domestic product GDP. The model is based on a closed economy, in which the total income or is consumed or is saved, and then reinvests in new capital. Therefore

$$R = s^R * Y \tag{2.18}$$

$$I = s^I * Y \tag{2.19}$$

$$u = \bar{u}, v = \bar{v}, w = \bar{w} \tag{2.20}$$

where $\bar{u}$, $\bar{v}$, $\bar{w}$, s^R, s^I are all exogenously given numbers between zero and one.

The tailor-made characteristic of FLOSS can be modelled by assuming that, the number of varieties depends positively on human capital, on the ICT-capital intensity, and finally on an exogenous term linked to R&D activity. Hence, the activities supported by PROPS and FLOSS are modelled in the following way:

$$\hat{A}^F = \psi_0^F * (\hat{h} + \hat{L}) + \psi_1^F * (\hat{K}_i^F - \hat{h} - \hat{L}) + \psi_2^F \tag{2.21}$$

$$\hat{A}^P = \psi_0^P * (\hat{h} + \hat{L}) + \psi_1^P * (\hat{K}_i^P - \hat{h} - \hat{L}) + \psi_2^P \tag{2.22}$$

where:

$$K_i^F = (1-w)(1-v)K_i \tag{2.23}$$

$$K_i^P = w(1-v)K_i \tag{2.24}$$

Note thay, ψ^i_j for $i = F, P$ and $j = 0, 1, 2$ are constant and non-negative parameters, and a hat over a variable denotes its instantaneous proportional growth rate. Finally, the production of human capital is modelled as:

$$\left(\frac{dh}{dt}\right) = \pi * (u * h)^{\gamma} * (v * K_i)^{1-\gamma} \tag{2.25}$$

where:

- π is a constant parameter reflecting the productivity of the human capital accumulation process;
- γ is a constant parameter between 0 and 1.

All these definitions aim to relate the conditional growth performance of this economy to the values of $\bar{u}$, $\bar{v}$, $\bar{w}$, $\bar{s}^R$, s^I, ψ^F, ψ^P and q, where q represents the difference in quality between FLOSS and PROPS use. The authors use the symmetry implied by 2.13, that is all users belonging to a certain group of software users would produce exactly the same amount of output, and hence users use exactly the same amount of raw capital x per activity. Therefore, output would be maximised by requiring that:

$$\left(\frac{\delta Y}{x^F_i}\right) = \left(\frac{\delta Y}{x^P_i}\right) \tag{2.26}$$

and hence requiring that:

$$\left(\frac{\delta K^e}{x^F_i}\right) = \left(\frac{\delta K^e}{x^P_i}\right) \tag{2.27}$$

and

$$x^P_i = q^{-\beta/(1-\beta)} * x^F_i \tag{2.28}$$

After some mathematical manipulations:

$$K = A^F * x^F + A^P * x^P = (A^F + q^{(-\beta)/(1-\beta)} * A^P) * x^F \tag{2.29}$$

$$x^F = (A^F + q^{-\beta/(1-\beta)} * A^P)^{-1} * K = \varphi^F * x^F \tag{2.30}$$

$$x^P = K * q^{-\beta/(1-\beta)} / (A^F + q^{-\beta/(1-\beta)} * A^P) = \varphi^F * x^F \tag{2.31}$$

the authors obtain:

$$K^e = K * (A^F * (q * \varphi)^F + A^P * (\varphi^P)^{\beta})^{1/\beta} \tag{2.32}$$

The authors simulated the model numerically forwards in time, by shocking the values of the exogenous variables u, v, w, s^R, s^I, ψ^F, ψ^P and q. They use a numerical method, because obtaining a closed form analytical solution is impossible.

To this aim some of the lesser-known or even unknown structural parameters of the model have been guesstimate, while others have been calibrated. Even a sensitivity analysis has been performed, to find out about the signs and orders of magnitude of the growth

effects associated with changes in the various system parameters.
Overall, the results highlighted the importance of human capital formation, and the way in which FLOSS can directly and positively influence the speed at which contributors to FLOSS communities collect new information and make disposal for themselves and for others.

Before concluding this brief introduction to the first part of the thesis, we describe other three works, which gave precious insights for the realization of our model. In the work [8], ***CRM Total Cost of Ownership: Fees, Subscriptions and Hidden Costs, CRM Outsiders and SugarSRM***, the authors provided a comparative price analysis of four leading CRM solutions for midmarket organization: Microsoft Dynamics CRM 2011, Sage SalesLogix, Salesforce.com, and SugarCRM.
Again in the work by Yankee Group [74], ***UnderstandingTotal Cost of Ownership of a Hosted vs. Premises-Based CRM Solution*** , the authors provided a comparative analysis about Total Cost of Ownership of a Hosted versus Premise-Based CRM solution, in particular they analyze the costs of SalesLogix and salesforce.com implementations.
Finally in the work by Sugar CRM [64], ***CRM Total Cost of Ownership Comparing Open Source Solutions to proprietary Solutions***, was analyzed the pricing options for CRM products. The analyzed products were five Sugar Professional, salesforce.com, NetSuite, Siebel OnDemand and Siebel CRM. Only one of them, Sugar Professional has a delivery model which includes source code. The authors aim to identify and compare the three generations of CRM implementations. In the first generation of CRM systems (Siebel), we have closed proprietary systems, expensive to purchase, customize and administer and difficult to install. In second generation, CRM vendors (salesforce.com) sell their CRM solution as a subscription-based service and hosted by the vendor. These solutions require a small upfront capital commitment and no infrastructure burden. The third generation of CRM (Sugar CRM Professional) blends the benefits of the two previous generations customizability, control, security, low or no initial capital expenditures, easy-on service model and upgradeability.

Chapter 3

On-Premise and On-Demand CRM Software Market Simulation Model.

In recent years the contractual relationships between software vendors and enterprise customers are exhibiting a deep transformation due to diverse factors, both economic and technological. Vendors cannot neglect the power of customers in software negotiation and their new perceptions. Sofware appears overpriced and the common thought is of paying a too high price for software which also supports low value business or unused processes. User willingness to pay is decreased and the opposition to the purchase of the upgrades is increased. Enterprise customers know that cannot have an agile enterprise with a traditional approach in software procurement. The opportunity to move from traditional pricing models, such as selling perpetual lincenses, to new pricing approaches has arrived. Pricing models based on periodic payments, in particular Software as a Service, are strongly gaining ground.

Nowadays, the term Software as a Service is everywhere, and is described as the future of software. SaaS, also called On-Demand software, is a software application delivery model. Providers of SaaS have got the possession of the phisical location, the hardware, and the system maintenance. Enterprise users access software via the Internet from anywhere, and at any time. Users usually pay a subscription and can run a single instance of a software in a robust infrastructure. Indeed, SaaS applications are delivered from a "multi-tenant" system. There is a single instance of software running and many individual or enterprise customers use this system for their own necessities. Together with SaaS, two other pricing approaches based on maintenance fees are also gaining ground. They are term licensing and commercial open source software. The first allows the use of software only for a fixed period, whereas the second distributes software without a license fee, but requires fees for support and maintenance, [18].

According to a Gartner survey:

”more than 95 percent of organizations expect to maintain or increase their investments in software as a service (SaaS), and more than one-third have migration projects under way from on-premises to SaaS.”

These data are extracted from a survey made in June and July 2011 on 525 organizations in nine countries spanning 12 vertical industries. The aim of this research was to understand the trends for SaaS adoption in enterprises. Enterprise users declared to have adopted

SaaS mainly for the easy and speed of deployment and for its cost-effectiveness, even if more than one-third of respondents complain, for example, the limited integration with existing systems and the network instability. The analysts found that the deployments of SaaS solutions vary greatly by industry. This is true both for horizontal and for vertical-specific SaaS solutions. For instance, currently 52% of communications industries, 51% of utilities industries, and 49% of banking and securities industries occupy the first places in a classification with respect to SaaS deployed. Moreover, this research declared that industries that are not using or considering SaaS solutions so far, are planning to use and consider SaaS solutions in a near future. In short, service oriented software is changing the world of software, and today new software pricing trends are a reality that we cannot disregard.

Among the 10 leading publicly-listed software vendors in US, Europe and Asia with respect to the presence in On Demand software segment, we can cite Google, Intuit, Microsoft, NetSuite, Oracle, Salesforce.com for US west coast; RightNow for US mid-west; Sage and SAP for Europe; and finally CDC for Asia, [62]. Among On Demand software applications offered by these 10 top vendors, Customer Relationship Management (CRM), Enterprise Resources Planning (ERP), Accounting, e-commerce, Human Resources Management (HRM), and Supply chain Management (SCM) can be reported.

In this Chapter a business model to analyse a software market in which vendors of perpetual licenses and vendors of SaaS compete is presented.

3.1 The Model proposed.

The main agents in software market are software houses and customers. We represent the firms, their creation, their life and all their business activities. We modelled two type of vendors: On-Premise and On-Demand vendors, highlighting their main characteristics according to the papers [56], [43], [1]. The first aspect to highlight is surely the way in which the two typologies of firms deliver their products [62]. On-Premise products are on site applications, instead On-Demand products are applications delivered from multi-tenant system via the Internet.

On-Premise vendors are only accountable for building and testing the software and for on-going support and bug-fixing; on the contrary, SaaS providers sell a complete service, which includes delivery, support and on-going maintenance. Hence the costs of the service delivering (costs of hosting, bringing customers, users onboard, cost of managing the application and data center environments), the costs of the on-going maintenance, and finally the costs of the updating play a key role in the success of these vendors. Indeed, these costs do not weight on the customer as in the traditional models, but they weight directly on SaaS providers.

SaaS providers must assure the integration to application level, to client level and to solution level, and moreover, they must guarantee customizable solutions as traditional On-Premise applications do. All this must be done without neglecting user satisfaction according to their service level agreements as regard uptime and performance.

Compared to perpetual licence vendors, who collect high up-front revenues, SaaS vendors collect low revenues during the course of the subscription (contract). For this reason, having efficient finance and accounting teams who provide stable financial processes is very important in this new pricing model. One important advantage of SaaS vendors with respect to traditional vendors is that they enjoy a lower ”time to market”. SaaS companies adopt an incremental R&D approach which must match to their revenue flow. Over time, the R&D will become a continuous-improvement approach. Time to market is lower because the deployment cycles are shorter and the new functionalities are marketed and delivered online. Another important advantage is that the capital invested for R&D is small. Indeed, the market response to the different products can be tested, and only after, the decision of investing large amount of capital can be taken into account. On the contrary On-Premise vendors invest large quantities of capital for R&D and develop and maintain multiple versions of a product to run on different platforms. In addition, SaaS vendors with recurring and predictable revenues can estimate their revenue flux and plan better their future business. It is clear that modeling a software house is not easy; there are many aspects to consider and many of them are very difficult to model. For this reason, implementing all the aspects characterizing a software vendor is impossible. In the next sections, we present in detail how we modelled a software vendor, its activities and its interaction with customers. Many are the semplification done, and so the products, the investment and pricing policies, and purchase choices of the customers modelled in this work, represent an ideal model which aimed to simulate the behaviour of the different agents in the complex system of the software market.

In the following, we present the modeling, simulation and implementation of the CRM software market. In particular, we modelled: traditional software or On-Premise vendors (OP), whose pricing model is that of the perpetual lincense, and On-Demand vendors (OD) which follow the Software as a Service pricing model [18].
All vendors produce vertical software products, which are substitutable. Substitutable goods are those goods which meet the same need, have the same functionality, but differ from each other as regard quality and price.
OP vendors produce a product which is formed by a primary and a secondary product. The primary products are vertical software products, instead, the secondary products are all the services of assistance, consultancy and maintenance associated with the use of the primary product. The secondary product is complementary to the primary, so the purchase of the primary cannot be separated from the acquisition of the secondary one.
On-Demand software vendors (OD) produce a product as a service, and a separation between primary and secondary products does not exist. They offer the customers a product accessible through the Internet and bear the hardware and network expenses, acquiring them by Cloud providers. Customers evaluate the purchase of a product through an utility function. They can acquire an OP primary product by paying a license fee, and a secondary product through a subscription, or a SaaS product on a subscription basis.

3.1.1 On-Premise and On-Demand vendors.

The modeled On-Premise companies are characterized by an initial capital. They work alone, and develop their own product, which differs in quality $Q(t)$ and price $P(t)$, when compared to others. The primary product is characterized by a price equal to $P_p(t)$, paid by consumers at the moment of the purchase. The secondary product is characterized by a price equal to $P_s(t)$, paid by users monthly.
On-Demand companies are characterized by an initial capital, and develop their own product, which differ in quality $Q(t)$, and price $P(t)$, when compared to others. In the case of On-Demand products, there is no distinction between primary and secondary product. The marketed product is only one, as mentioned in the previous section. It is a complete product that includes delivery, support and on-going maintenance. The pricing model adopted by

these vendors entails periodic payments on a monthly subscription base. Enterprise customers pay a monthly fee based on usage (the number of people at the company using the software).
The quality of the products has been defined as in the Haruvy, Sethi e Zhou's work [16], and human capital as in the economic model, presented in a Rishab Aiyer Ghosh's work [52]. The quality is defined by a differential equation.

Therefore, the quality of the products at time step t is defined as:

$$\left(\frac{dQ_i}{dt}\right) = h_i(t)N_i - \delta Q_i \tag{3.1}$$

where:

- N: number of developers of $i-th$ firm, who work at the primary or secondary product, or SaaS;
- δ: quality depreciation rate;

For quality we mean modeling the cumulative effects of all the software features that contribute to the perceived value of the software product. So, equation 3.1 models *Functionality*, *Reliability*, *Efficiency*, *Usability*, *Maintainability*, and *Portability*. The six characteristics listed are those established in the ISO/IEC 9126-1 standard and they characterize and define the quality of a software product.
We define the per-capite human capital h, of $i-th$ firm at time t, as in the work of Rishab Aiyer Ghosh [52]. hd is equal to the productivity per-capite, and is given by:

$$\left(\frac{dh_i}{dt}\right) = [\pi * (u * h_i)^{\gamma} * (v * C_{cum/PerCapite,i})^{1-\gamma}] \tag{3.2}$$

where:

- π: is a constant parameter reflecting the productivity of human capital accumulation process;
- u: is the fraction of time spent on human capital formation h;
- γ: weighs the fraction of human and ICT capital in human capital accumulation process;
- C: is the capital per employee invested in the quality of the product;

In the case of OP firms, N is set to the total number of employees of the company, assuming: $N_p = \frac{2}{3}N$ and $N_s = \frac{1}{3}N$.

All the values of the parameters present in these equation are reported in Table 3.2.

3.1.2 Investment policies.

Each company has an amount of initial capital $A_i(0)$ depending on the number of employees, which do not vary during the simulation, and enters the market with an initial investment $C_i(0)$. According to our experience, for On-Premise companies, the initial capital available has been set equal to 50,000.00 per employee.
Instead, On-Demand vendors have an higher amount of initial capital than OP vendors. This because they must invest in a complete product, and so they need both hardware and software. It can be assumed that OD firms' initial capital is equal to 80,000.00 per employee.
To enter the market, every firm invests at the initial time $t = 0$ a fraction β of its capital $C_{initial,i}$, with β given by a normal variable with average and standard deviation set so that its value fall in the range [0.8,1] with a probability equal to 99%.

$$C_i(0) = A_i(0) * \beta_i \tag{3.3}$$

Supposed that N_{prim} is the number of work unit engaged for the primary product, N_{second} is the work unit number engaged for the secondary product, s is the per-capite wage of the firm's developers, for OP firms the capital invested in quality $C_{Q,i}(0)$ at the initial instant $t = 0$ by the $i - th$ firm is given by:

$$C_{(Q,i)}(0) = C_i(0) - s(N_p + N_p) \tag{3.4}$$

where

$$s(N_p + N_s) = C_{wage}(0) \tag{3.5}$$

is the capital invested in wages at initial time $t = 0$ and each month.

In the case of OD firms, a fraction Y of the initial capital invested is assigned to cover the hardware and network expenses, so the capital invested in quality is given by:

$$C_{(Q,i)}(0) = C_i(0) - Y_i * C_i(0) - s(N_p + N_p) \tag{3.6}$$

where:

$$C_{HardwareNetwork,i}(0) = Y_i * C_i(0) \tag{3.7}$$

is the capital invested in hardware and network expenses.

Now, we look at the investments at generic time instant t, particularly for the updating of products. Concerning OP firms, they update their products at time intervals $\Delta_{On-Premise,i}$ equal to 18 months [?]. For these firms the invested capital in quality, $C_{Q,updating}(t)$, is a normal variable equal to a fraction m of the capital invested at time $t = 0$:

$$C_{Q,updating}(t) = m_i(t) * C_{(Q,i)}(0) \tag{3.8}$$

Note that, the prices of software updates are included in the maintenance costs, as it usually happens [10].

Finally, at the smaller time intervals, equal to 1 month, these firms invest smaller amounts of capital to maintain product quality to an acceptable level. We assume that this investment is much smaller than the investment made for producing it. In particular we have:

$$C_{(Qmonthly,i)}(t) = r_i(t) * C_{Q,i}(0) \tag{3.9}$$

The parameters m and r, in the equations above, are normal variables defined as reported in Table 3.2.

In the case of On-Demand vendors, the updating process is incremental. Small amounts of capital are invested continuously. In particular, a quantity equal to $C_{(Qmonthly,i)}$ is monthly invested for maintaining product quality to an acceptable level, and for maintenance of the product. In addition, a quantity equal to $C_{(HardNetwMonthly,i)}$ is invested monthly for hardware and network expenses. Both these quantities are proportional to a fraction of the capitals invested at the initial time.

$$C_{(Qmonthly,i)}(t) = [r_i(t)] * C_{(Q,i)}(0) \tag{3.10}$$

$$C_{(HardNetwMonthly,i)}(t) = w_i(t) * C_{HardwareNetwork,i}(0) \tag{3.11}$$

All investments made by each company are equally divided among each employee in the company, and increase the developers' productivity h.

Naturally, companies invest only when their annual financial statements allows it. The annual financial statement defines the company's financial situation. It is composed of two documents: balance sheet and profit and loss account. The former defines the economic activity of the firm, the latter defines its financial and patrimonial position. These two documents correspond to two fundamental equations, identified as BS and PLA and defined respectively as:

$$BS : Assets = Liabilities + NetCapital \tag{3.12}$$

$$PLA : Revenues = Cost + Profit \tag{3.13}$$

The profit is the element that connects the profit and loss account to the balance sheet. In fact, the total of the activities will be equal to the total of liabilities only if we add the profit to net capital, that is the difference between revenues and costs.

In our model these equations are defined as:

$$BS_i : Assets = [A_i(0) - C_i(0)] - C_{Qupdating,i}(t) - C_{(Qmonthly,i)}(t) - \\ C_{(HardNetwMonthly,i)}(t) - C_{wage,i}(t) - I_i(t) + Profit_i(t) \tag{3.14}$$

$$PLA_i(t) : R_i(t) = -C_{Qupdating,i}(t) - C_{Qmonthly,i}(t) - C_{wage,i}(t) - \\ C_{(HardNetwMonthly,i)}(t) - I_i(t) + Profit_i(t) \tag{3.15}$$

The term $R(t)$ represents the revenue of $i-th$ company at time t, $Profit_i(t)$ the profit, and $I_i(t)$ the instalment of the bank loan. Note that, the difference between $A_i(0)$ and $C_i(0)$ remains as cash. The value of the revenue $R(t)$, at time t for $i-th$ company, is given by:

$$R_i(t) = [P_{i,OP/OD}(t) * N_{c,i}(t)] \tag{3.16}$$

where:

- $P_{i,OP/OD}$ is the price of the OP or OD products respectively, and
- $N_{c,i}$ is the number of users at time t and for the firm $i-th$.

Therefore, companies can make a new investment only if the budget allows it, that is only if the following condition is verified:

$$-C_{Qupdating,i}(t) - C_{Qmonthly,i}(t) - C_{wage,i}(t) - C_{(HardNetwMonthly,i)}(t) - I_i(t) + Profit_i(t) * (1 - v_{i,1}) > 0 \quad (3.17)$$

In equation 3.17, $v_{i,1}$ indicates the percentage of profit allocated to the charter members, and so it is written off to feed a debt toward such subject as dividends; only the remaining part can be invested again. When equation 3.17 goes to zero or becomes smaller than zero, the company can request a bank loan, if this loan exceeds its initial capital the company goes bankrupt. This constraint is applied only to the firms active in the market for at least six months.

3.1.3 Pricing policies.

The companies can define the price as a function of their costs, using a demand- driven approach, or using a competition- oriented approach [61]. The more suitable approaches are the last two, but applying them is not simple. The companies would have to know customer preferences and competitors' behaviour [48], and this information is not easy to obtain. Software vendors adopt pricing strategies fundamentally different from those of other industries [61]. Software products do not lose quality as a result of their usage, even if there is a loss of value over time. Moreover, a high production cost is needed to produce the first copy, but a low production cost is need for the next copies. Another feature of software products is the existence of network effects. The value of software depends on its properties, but also on the number of users who adopt it. The larger the number of users, the better it is for the users. Furthermore, the network effects have a significant impact on pricing strategies of software vendors, who adopt strategies to hinder the market entry for competitors and bind the customers to the current provider, with high switching costs.

To model the pricing strategies, we propose a pricing mechanism which vary over time and for the two type of vendors, OP and OD. At the time t_{entry}, when the firms enter the market, the following pricing rules are assumed:

- for On-Premise vendors:
  -
  $$P_{OP,i}(t_{entry}) = [1 + \o_i] * \left(\frac{N_{fTOT}(t_{entry}) C_{invested,i}(t_{entry})}{K_1 * N_{uTOT}(t_{entry})} \right) \quad (3.18)$$
  - This price is a license fee per end-user, where $C_{invested,i}$ is the investment made by company, N_{fTOT} the number of companies on the market, N_{uTOT} the total number of customers in the market, and $\o_i$ the percentage of profit.
- for On-Demand vendors:
  -
  $$P_{i,OD}(t_{entry}) = [1 + \o_i] * \left(\frac{C_{initial,i}(t_{entry})}{k_1 * N_{uTOT}(t_{entry})} \right) \quad (3.19)$$

– This price is a fee per end-user/month, where $C_{initial,i}$ is the initial investment made by the company.

In both equations the parameter K_1 models the amortization period of the production costs of the software. The equations above reported define the price at time in which the firm enter the market. Instead, over time the companies determine the price of their products following the mechanism described below. As in the work by Rohitratana and Altmann [30], we defined a pricing mechanism called *Derivative-follower pricing scheme*, in which vendors decide to increase or decrease, or not vary their price as a function of their revenues, and neglect the market condition.
The pricing scheme adopted is the following: vendors evaluate their revenue, if the revenue in $t+1$ is higher than the revenue in t of at least 5% vendors do not vary their pricing mechanism; otherwise, if the revenue in $t+1$ is lower than the revenue in t of at least 5% vendors decide to adopt one of the following possibility:

1. keep the price constant ;
2. increase the price of a quantity σ;
3. decrease the price of a quantity σ;

where σ is a constant which depends on the type of vendors: $\sigma_{OP} = 30$, and $\sigma_{OD} = 2$.

The time trend of revenues is evaluated every three months, and the chosen pricing mechanism is maintained for six months, before a different pricing strategy can be applied. We assumed that all the prices set can never be lower than a threshold P_{Th}, defined as:

$$P_{Th}(t) = \left(\frac{N_{fTOT}(t) * C_{invested,i}(t_{entry})}{K_2 * N_{uTOT}(t)}\right) + \left(\frac{N_{fTOT}(t) * C_{invested,i}(t) * 12}{K_3 * N_{uTOT}(t)}\right) \tag{3.20}$$

The first term models the amortization of production costs, the second term models the amortization of the annual costs of software maintenance. All the parameters K_n (in the equations 3.18, 3.19, and 3.20) could be calibrated setting them equal to the years in which the firms expect to have a positive return on investments. The amortization period must be determined on the basis of an amortization plan, taking into account the useful life of the software to be adopted and its cost.
Naturally, the pricing mechanism adopted in our model is a semplification of the real pricing mechanism, and aims to obtain the same price's values as the real prices. For the parameters K_n of OD firms, we assumed higher values than those of OP firms. This because, in case of SaaS vendors, low revenues are collected during the course of the subscription (contract), and consequently having a long amortization period becomes very important.

Note that, customers are enterprise customers, that buy software for a number of users varying between 100 and 1000. The price calculated from our model is a price per end-user. The usage of software can vary over time, but in OP case the enterprise users pay according to the number of end-users, in SaaS case pay according to pay as you go model, and so the model calculates the real usage of software. The enterprise customers pay for the number of end-users who may use the software. We assumed that, this number varies in the range between $(s - 0.3 * s)$ and s, assuming s equal to the installation size of end-users.

As above affirmed, we modeled a vertical software market, and the prices modeled are formed regulating all the parameters of the model, so that the prices modelled with equations 3.18 and 3.19 follow the order of magnitude of the prices for hosted and On-Premises-Based CRM solutions presented in [60], [10], [74], [8]. Therefore, in the OP pricing mechanism, we also included a discount mechanism as in the work [8]. In the work [8], the authors provide a comparative price analysis of four leading CRM solutions for midmarket organization, defined by Forrester Research as organizations with revenues of less than $1 billion and fewer than 1,000 employees. The CRM solutions included in this analysis are Microsoft Dynamics CRM 2011, Sage SalesLogix, Salesforce.com, and SugarCRM.

In the work by Yankee Group [74], the authors provide a comparative analysis about Total Cost of Ownership of a Hosted versus Premise-Based CRM solution, in particular they analyze the costs of SalesLogix and salesforce.com implementations.
To understand the orders of magnitude of the real costs, we briefly review some prices reported in these two works. In [8], for example, SugarCRM (CRM OD solution) is offered in four different editions, professional, corporate, enterprise and ultimate, with a price that varies between $30 (monthly user fee) for Sugar Professional and $250 (monthly per user) for Sugar Ultimate. The authors made also a pricing analysis about on premise Sage SalesLogix, in which the prices of primary proprietary products varied with the number of users. The base price can vary between $795 and $1,095, but the users have a 12% discount when they buy more than 50 licenses, and a 24% discount for purchases over 200 copies.

To quantify the prices for the secondary product, the server software and the hardware, we used some data presented in [74]. So, in the On-Premise case, the customers support:

- a cost P_s equal to 20% of P_p,
- a cost $P_{S/H}$ equal to 57% of P_p,
- a monthly cost $P_{S/H/monthly}$ equal to 24% of $P_{S/H}$.

where

- $P_{S/H}$: are the costs for hardware procurement and selection, and web/wireless servers software,
- $P_{S/H/monthly}$: are the costs for ongoing operational support.

For what concerns the upgrades, their prices depend on how much customization and integration of the application had made to the software application. They can vary from $50,000 to $200,000 [74]. Moreover, it is not easy to compare OP with OD solutions [74] because upgrades in OD solutions occur on average 3 or 4 times a year and then cost is minimal. For this reason, we assumed that these prices are included in the secondary price in the OP case, and in the monthly fee in the OD case.
To compute the cost of switching P_{Sw}, we proceed according to data in work[74]:

- $P_{Sw/OD-OP}$ is the sum of costs $P_{S/H}$, of $P_{C/I/OD-OP}$, equal to 28% of P_p, and of cost P_p,
- $P_{Sw/OP-OP}$ is equal to cost $P_{C/I/OP-OP}$, equal to 28% of cost P_p,
- $P_{Sw/OP-OD}$ is the sum of cost $P_{C/I/OP-OD}$, equal to 10% of P_{OD}, and of cost P_{OD} itself,
- $P_{Sw/OD-OD}$ is equal to cost $P_{C/I/OD-OD}$, equal to 10% of cost P_{OD}.

Where:

- $P_{C/I}$ is the cost for basic customization and integration: "IT customizes business functionality using development tool to configure interface and underlying data objects. Create and modify user interface. Create and modify custom reports. Create database schema. Deploy to servers and users desktops. Resolve synchronization issues. Integrate with legacy systems" [74].
- $P_{S/H/monthly}$ is the cost for ongoing operational support divided in:
  - Basic Data Center Operations: "Monitoring and backup, including adding application servers to the intrusion detection system, configuring monitoring software and building tape backup facility" [74].
  - General Patches, Performance Tuning, Basic UI Refinement:"Apply maintenance patches and upgrades to multiple machines. Add, modify user accounts. Troubleshoot client desktop issues for remote users Troubleshoot performance issues. General maintenance of multiple servers (application servers, Web servers, synchronization servers, database servers)." [74].

All the assumptions done, in this and in previous sections, aim to simplify the real business policies, which otherwise would be intractable due to the limited quality and quantity of data available. For this reason, from the work [74], we extracted approximate data, which attempt to represent in as realistic a way as possible the real time trend of prices in the software market.

3.1.4 Customers.

In our model customers are enterprise customers. The number of customers in the market is equal to N_C. Every customer buys software for installations that vary between 100 and 1,000 end-users. So, we decided to study small and medium enterprise customers [8]. Each customer has its own portfolio, varying over time. In particular, we assume that $j-th$ customer has an initial budget and a monthly income M. The initial budget and its updates vary with the number of end-users and over time. This portfolio indicates the customers' willingness to pay for the ownership of a software product. When it goes to zero, the customers' willingness goes to zero, and consequently the customer does not buy products any more.

As in reality customers can compare their products to other products in the market, to verify if their current product is the most convenient choice. Logically, not all customers evaluate the purchase of a new product at the same time. For this reason, at each time t, a random number of customers is drawn to re-evaluate their purchase choice.
Initially, a customer evaluates all products on the market and chooses a product using an utility function and buys it. In this way it binds itself to a vendor. At each time step customers drawn look for the best product in the market, comparing the values of the utility functions associated to their products with those of all the other products in the market, and in the end they decide whether or not to replace their products.
In general, the utility function UF indicates the subjective evaluation about the attitude of a product or service to meet an economic need (use value), and depends on: quality, price, inclination towards new pricing trend, switching costs ρ, network effects ϵ, and on a multiplicative normal noise χ.

For istance, for j-th customer and for the primary product of $i-th$ firm, UF is defined as:

$$UF_{j,i}(t) = a_Q * Q_{i,p}(t,\chi) - a_P * P_{i,p}(t,\chi) + \eta(\chi,t) + a_{Ne} * \epsilon_{j,i}(t,\chi) - a_{Sc} * \rho_{j,i}(t,\chi) \quad (3.21)$$

The first two terms take into account the quality and the price of the products. The third term, η, models the customer feeling toward new Cloud technology. So, it models how customers evaluate the benefits and the risks associated to SaaS adoption, as illustrated in section 3.1. The values of η vary with the size of enterprise customers according to Slowinski et al. [62] and Etro [17]. They affirm that the greatest potential growth in SaaS comes from the smaller business areas. Indeed, in this area the companies invested in software less than large enterprises, and are generally running outdated solutions. Consequently, they see in cloud solutions a way for increasing their competitiveness without facing the high up-front costs typical of On-Premise solutions. For this reason, we defined η as follows:

- for customers with a number of end users higher than 500:
  - $\eta \in Norm(0.1, 0.02)$
- for customers with a number of end users lower than 500:
  - $\eta \in Norm(0.2, 0.02)$

Concerning the network effect term ϵ, each customer assigns a value to it as a function of the number of customers which have the same product. It describes a situation in which the utility that a customer derives from the consumption of a product depends (positively or negatively) on the number of other customers who consume the same product. Indeed, the larger the number of customers of the $i-th$ firm, the better for them.

The network effect is defined as follows:

- if $N_{c,t} < N_{c,M1}$
$$\epsilon = 0$$
- if $N_{c,M1} <= N_{c,t} <= N_{c,M2}$
$$\epsilon = \left(\frac{(N_{c,t} - N_{c,M1})}{N_{C,M2} - N_{c,M1}}\right)$$
- if $N_{c,t} > N_{c,M2}$
$$\epsilon = 1$$

where:

- $N_{c,t}$ is the number of customers at time $t > 0$,
- $N_{c,M1} = a * N_{c,TOT}$ is the threshold above which the network effect starts,
- $N_{c,M2} = b * N_{c,TOT}$ is the threshold above which the network effect saturates to 1.

If $N_{c,t} < N_{c,M1}$ the number of customers is too low, and there is not network effect; on the contrary, if $N_{c,t} > N_{c,M2}$ the software usage is very spread and network effect saturates to 1.

The random term χ is a multiplicative normal noise, with average 0 and standard deviation 0.01. It perturbs the generic term x, in equation 3.21, following this equation:

- if $\chi > 0$ then
  -
$$x' = x * (1 + \chi)$$
- if $\chi < 0$ then
  -
$$x' = \left(\frac{x}{1 + |\chi|}\right)$$

The latter term in the equation 3.21 accounts for the swicthing cost ρ. Note that, the values of quality, price and switching costs must be normalized. To this purpose, we proceeded in the following way. For instance, assuming that the minimum price is P_{min} and the maximum price is P_{max}, the normalized price C_i function of the generic price P_i, is given by the following equation:

$$C_i = \left(\frac{P_i - P_{min}}{P_{max} - P_{min}}\right) \tag{3.22}$$

In this way all the quantities in the utility function are in the range [0,1].

All the terms in the utility function are customized for each customer through the a_j coefficients, characterized by a normal distribution. Their values are given by:

- $a_Q \in Norm(0, 0.1)$
- $a_P \in Norm(0, 0.1)$
- $a_{Ne} \in Norm(0, 0.1)$
- $a_{Sc} \in Norm(0, 0.1)$

So the utility function values, calculated by the $j-th$ customer for each $i-th$ firm, are the sum of the features of the $i-th$ products weighed by the coefficients a_j and by the random term χ. In this way each user has a different perception of the products in the market, with respect to the other customers.

3.2 A simulation-based approach to solve the proposed model.

Our model represents a vertical software market, that is a segment of the software market in which some applications compete, giving functionalities to perform a specific job. These are applications dedicated to a specific manufacturing or services sector (for example, footwear, pharmaceuticals, food, metalworking, or credit), in contrast with the horizontal applications that are general-purpose software. The vertical software market studied in this work is the CRM maket. CRM is now a successful business strategy: a new method of work and processes management, which through the achievement of organizational efficiency allows firms to increase the company's turnover while ensuring a high level of customer satisfaction. A CRM provides a new approach that puts the customer, and not the product, at the center of the business. The CRM is used by firms to identify and manage the profiles of acquired and potential customers, as well as to develop activities and strategies which on the one hand help to capture new customers and on the other to maximize the satisfication of faithful customers, trying to understand their needs and expectations.

Using to the data reported in [**?**], to [25] and [27], we can estimate the number of thousand CRM customers for each firms: 100 for SAP; 79 for Oracle; 52 for Salesforce; 31 for Microsoft; 24 for Amdocs; 194 for others, the total number of customers in the CRM market being 490.

In the followings, we use a simulation-based approach to analyze the software market, and in particular the CRM software market. It is a complex dynamic system, characterized by a set of separate entities, which interact through interdependence relationships or reciprocal connections. Its characteristics can not be obtained through the sum of the parts which constitute it. The network of relationships between the entities produces non-linear effects, which cannot be explained by studying each component individually. The presence of non linearities forced us to adopt the simulation as an alternative to analytical models in the study of complex dynamic systems.

In mathematics, a specific formalism for the description of dynamic processes differential equations yielding the evolutionary trajectory of the system as a function of time. The analytical methods often require the adoption of generalized descriptions for the behavior of the model, or the building of complicated systems of differential equations, capable of describing different behaviors for different intervals of the state variables. On the contrary, with a simulation model we are able to represent the system in an easier way, studying its aggregate behavior. The simulation is a methodology which is part of the so-called experimental mathematics [49]. It is a representation of the system realized through a computer system. Through programming languages, to define the properties of a system in detail is possible, determining the behavior dynamically, as function of its current state. Using simulation, to introduce in simple way a conditional behavior through conditional constructs is possible, allowing us to analyze and formalize complex systems, which are otherwise intractable. Besides these advantages, the simulation issues are the generalization of the results. It does not provide the same quality and information content as an analytical solution. In fact, while the analytical solution provides full information on the system represented, the simulation is only able to provide information on individual instances of a possible future path of the model, which is often determined by the initial conditions.
Taking into account all the advantages and disadvantages of a simulation-based approach and given the limited quality of the data available, it makes sense to analyse the business trends in a ideal CRM market reproduced by us through the ***Base Run*** (*set 1*), where we use parameter values which can be considered broadly correct. To analyse the model proposed in the previous sections, we studied the competition among OP and OD firms, analysing the behaviour of the market over a simulation period equal to 96 months.

3.2.1 Base Run results.

In the ***Base Run*** (*Set 1*), we run the model setting all its parameters as reported in Tables 3.1, 3.2, and 3.3. The values reported are taken from the literature, from market analysis and

Table 3.1: *Parameters of the model related to the customers.*

Description of parameter	Value
Initial number of customers on the market, N_C:	$N_C = 40,000$
Number of end users for each enterprise customers:	Characterized by a normal distribution with average $\mu = 550$ and standard deviation $\sigma = 225$.
Initial Budget of each users per end-user:	Characterized by a normal distribution with average $\mu = 30000$ and standard deviation$\sigma = 5000$.
Increment of the initial budget per end-user:	Characterized by a normal distribution with average $\mu = 15000$ and standard deviation $\sigma = 2500$.
Number of customers drawn at each time instant for evaluating the possibility to change products:	Equal to $N_C/12$ of the number of users.

Table 3.2: *Values of the parameters of On-Premise and On-demand vendor equations.*

Parameter	Description	On-Premise Vendors	On-Demand Vendors
N	Number of developers for each firm:	Characterized by a normal distribution with average $\mu = 135$ and standard deviation $\sigma = 21$	Characterized by a normal distribution with average $\mu = 165$ and standard deviation $\sigma = 11$.
$A_i(0)$	Initial capital available to each company:	Equal to 50,000.00 per employee.	is equal to 80,000.00 per employee
Y_i	Fraction of capital invested for hardware expenses:	–	Characterized by a normal distribution with average $\mu = 0.45$ and standard deviation $\sigma = 0.02$.
r_i	Fraction of capital invested monthly:	normal variable with average $\mu = 0.002$ and deviation standard $\sigma = 0.00033$ in equation 3.9.	Normal variable with average $\mu = 0.003$ and standard deviation $\sigma = 0.00033$.
ω	Fraction of capital invested monthly in hardware and network expenses:	–	Normal distribution with average $\mu = 0.0004475$ and standard deviation $\sigma = 0.000025$.
$m_i(t)$	Fraction of capital invested for updating the products:	Characterized by a normal distribution with average $\mu = 0.005$ and standard deviation $\sigma = 0.00033$.	–
$\Delta_{OP,i}$	Interval of updating of $i-th$ OP firm:	18 months.	
K_1	Parameter which depends on the amortization years of the production costs	60	240
K_2	Parameter which depends on the amortization years of the production costs	60	240
K_3	Parameter which depends on the amortization years of the annual costs for software maintenance	12	120

some of them are set in order to obtain prices that match the real prices in the market..
We analyzed the competition between five OP firms, which enter the market at $t_{entry} = 0$, and six OD firms, which enter the market in pairs, at $t_{entry} = 10, 20$, and 30. This matches the real world, where the entry of OD firms occured after that of OP firms. The results obtained in this first set are reported in the Figures 3.1 - 3.4, and in Table 3.4.

The figures show the time trends of quality and prices, for the two kinds of products, OP and OD; instead, the table shows the number of customers at $t = t_{entry}$ and at $t = T$ for the different firms in the market. This data highlights how OD CRM firms are able to conquer quite a big market share and to compete with OP firms.
The values of OP CRM is bigger than that of OD CRM, in agreement with the fact that OP CRM solutions are more customized than OD CRM ones [47]. Indeed, an OP CRM solution

Table 3.3: *Parameters common to both types of vendors.*

Parameter	Description	Value
β_i	Fraction of the initial capital invested by $i-th$ firm:	Characterized by a normal distribution with average $\mu = 0.9$ and deviation standard $\sigma = 0.03$.
$ø_i(t)$	Profit that $i-th$ firm want to obtain by the sale of products:	Characterized by a normal distribution with average $\mu = 0.15$ and deviation standard $\sigma = 0.02$.
$\upsilon_{1,i}(t)$	Percentage of profit allocated as dividends to the charter members:	equal to 0.5.
δ	Quality depreciation rate, (in equation 3.1):	0.03. (Haruvy, Sethi and Zhou [16])
π	Productivity of human capital accumulation process:	0.025. (Ghosh, p.237 [52])
u	Fraction of human capital accumulation process:	0.1. (Ghosh, p.237 [52])
π	Constant parameter reflecting the productivity of human capital accumulation process:	0.025. (Ghosh, p.237 [52])
u	Fraction of time spent on human capital formation h:	0.1. (Ghosh, p.237 [52])
γ	Fraction of human and ICT capital in human capital accumulation process:	0.9. (Ghosh, p.237 [52])
s	Wage of developers:	3000

allows for much more control of the customizations, performance and upgrades than OD solution. It is necessary to hjghlight that the quality values are very high, because of the chosen initial values for the human capital per-capita and the quality per-capita. Anyway, all these considerations do not change if we assume for these parameters lower values, because they are accounted as normalized values in the utility function.
In addition, the quality values also depend on the capital invested by each firm to produce, improve and maintain the product. The investment policies, defined in this work, stem from analysis of the market and our experience. However, it is not easy to know the specific values of the investments done by the firms. For this reason, most of our assumptions aim to calibrate the whole model in order to obtain realistic magnitude orders of the prices. With this assumption, we accept any value for the investments, also if they do not correspond to realistic values.
The results reported in this section aim to highlight the weaknesses and the potentialities of our model, and show how it can become an useful tool to analyze real past business strategies and find a winning business strategy.

As already said, some of the parameters have been calibrated in order to obtain realistic prices for the products. For this reason, we set the value of the initial capital per employee equal to 50,000 or to 80,000, respectively for OP and OD firms, and the parameters Y_i, r_i, ω_i, m_i, and k_n as indicated in Table 3.2.

As a result of our simulation the prices of OP CRM vary between about 300 and 1,000 for primary products, and between about 80 and 260 for secondary products; the prices of OD CRM vary between about 180 and 350, matching the real prices in the market.

Note that, at time $t = t_{entry}$ a number of users equal to $\left(\frac{N_C}{3}\right)$ is drawn to evaluate the new products entering the market, in order to model the interest provoked by the entry in the market of new products. At other every times the fraction of customers drawn to re-evaluate their purchase choices is equal to $\left(\frac{N_C}{12}\right)$.

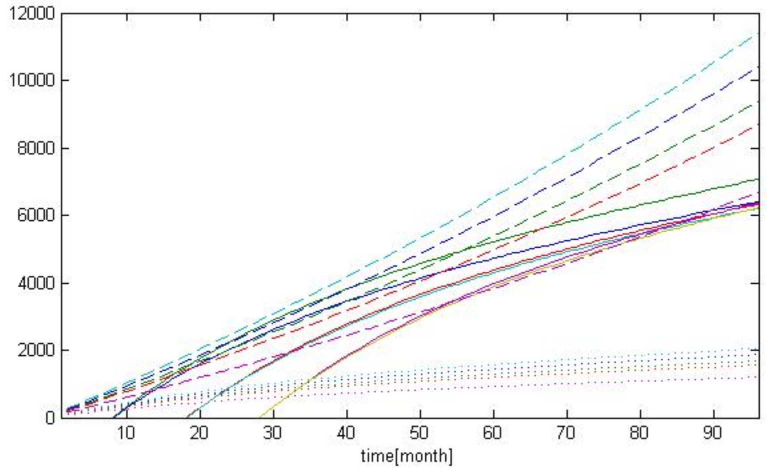


Figure 3.1: Quality of the OP primary (dashed lines), OP secondary (dotted lines) and SaaS (solid lines) products (***Base Run***).

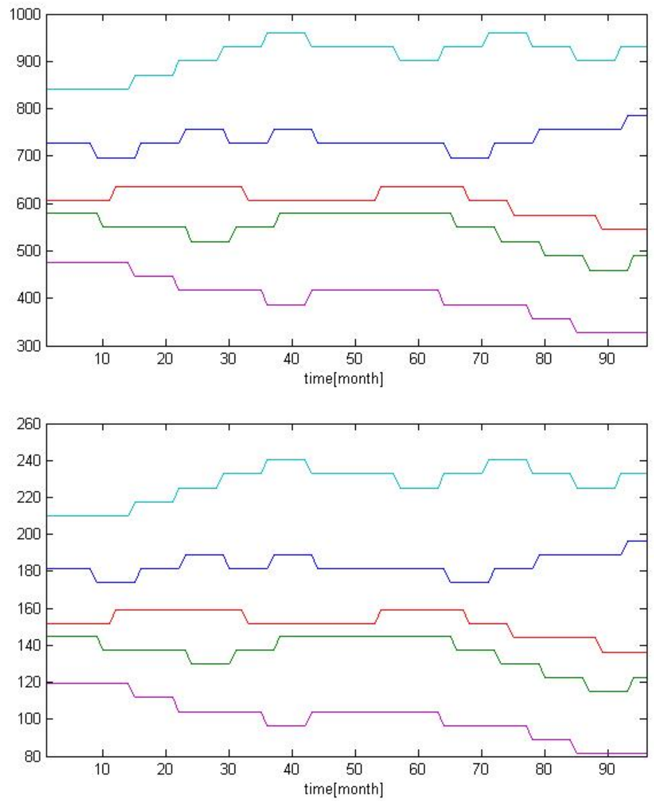


Figure 3.2: Price of the OP primary and secondary products (***Base Run***).

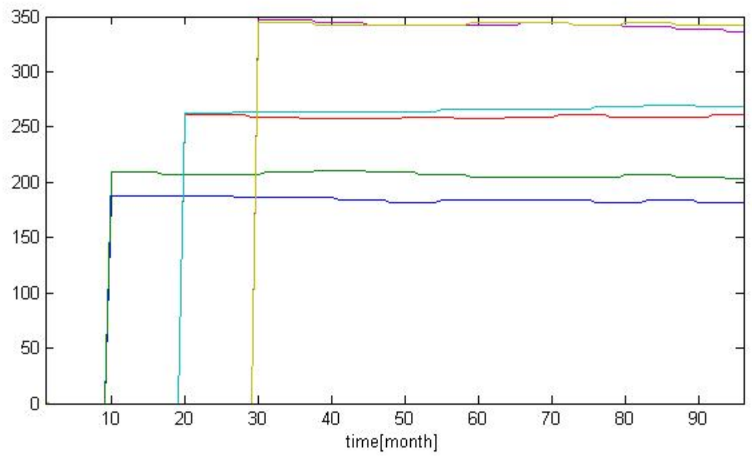


Figure 3.3: Price of SaaS products (***Base Run***).

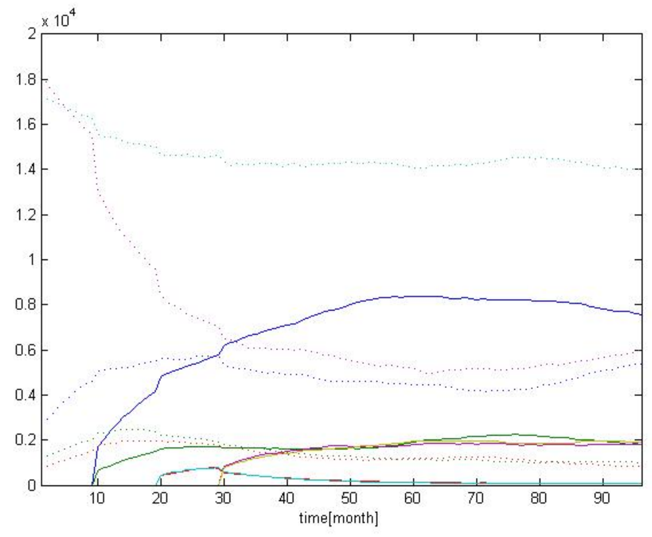
Table 3.4: *Number of customers in the different firms in the market for the* ***Base Run****.*

Set 1			
OP customers		OD customers	
at $t = t_{entry}$	at $t = T$	at $t = t_{entry}$	at $t = T$
2,657	5,361	2,055	7,578
1,219	1,009	769	1,890
713	831	481	58
17,259	14,018	509	69
18,152	5,981	931	1,789
		876	1,913
Total number of Customers.			
40,000	27,200	5,621	13,297

In figure 3.4, we analyze the time trends of the distributions of customers among the firms. OP firms, get a high market share in the beginning. Then, their customers to decrease when OD firms enter the market. All the firms are able to stay in the market, two OD firms stay in the market with very a small market share, namely with 58 and 69 customers only.
Let us highlight again that, due to the lack of experimental data, all parameters in this first set have been calibrated in order to obtain prices similar to those that nowadays we observe in the real software market. The market trends we obtain seem to reflect the real trends of the market. Few firms are able to survive with big market shares, whereas the remaining survive with much smaller market shares.

In particular, one OP firm gets a market share equal to 35% at $t = T$, and another OD firm gets a market share equal to 19% at $t = T$. Two other OP firms get market shares equal to 13% and 15%. All other seven firms together get a market share of only 18 percent.

Parameter χ plays a very important role in the determination of the market shares. Its definition is reported in section 3.1.4. It has been turned to the firms to stay in the market, allow most not to reduce substantially their number of customers. In the next sections we report a detailed sensitivity analysis.



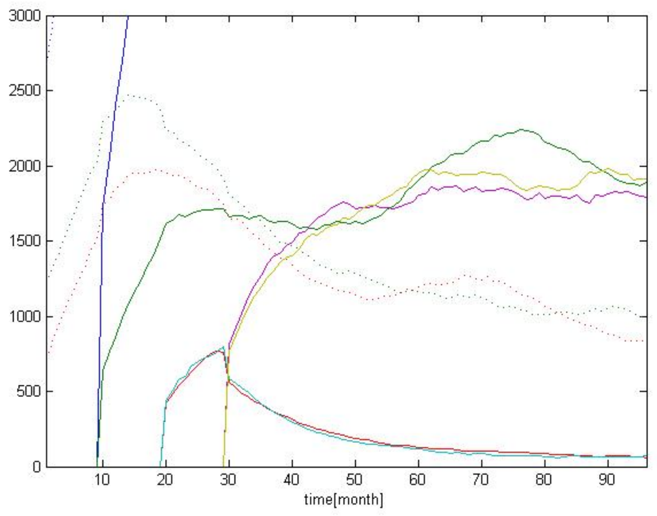


Figure 3.4: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (***Base Run***). The figure below expands y axis to highlight the customer distributions with smaller values.

3.2.2 Sensitivity analysis.

Sensitivity analysis of the model to the utility function parameters.

We start with the study of the sensitivity of the model to various terms in the utility function. The utility function value varies between 0 and 1, as any other term in this function. The factors Q_i, P_i and $\rho_{j,i}$ are constrained to this range following a normalization process, whereas all the other terms vary in this range by their definitions.
To study the influence of the parameters, χ, η, ρ and a_j, we designed the simulation sets numbered from *2* to *8*. In the followings, we describe the first four sets and the obtained results.
In *sets 2.i* we investigated the sensitivity of the model while decreasing the value of the random noise χ, starting from *set 2* to *set 2.2* (see Table 3.10). Finally, in *set 2.3* we set to zero the

parameter χ.
In *set 3*, we investigated the sensitivity of the model to the coefficients a_j. We decreased the values of the coefficients a_j with respect to *set 1*. We set them as belonging to the normal distribution Norm(0,0.05).

We run two sets, called *Set 4* and *Set 5* to investigate the sensitivity of the model to the network effects ϵ. In the first set, we repeated *Set 1* neglecting the network effects; in *Set 5*, we took the network effects into account, varying the parameters a and b. Remember that a defines the threshold above which the network effect starts, and b the threshold above which the network effect saturates to 1. In *Set 1*, a is equal to 0.2 and b is equal to 0.7; instead, in *Set 5*, a is equal to 0.1 and b is equal to 0.6.

Finally, *Set 5.1*, repeats *Set 5*, neglecting both network effects, and parameter χ, whereas set *5.2* takes network effects into account, but neglects parameter χ.
The results of sets *2.i* show that, if we neglect the multiplicative normal noise χ, we get a less uniform distribution of customers among the firms than the other sets, (see Tables 3.5 and 3.6). In fact, in sets *2.i* there are firms with no customers, or with a very low number of customers. Decreasing the value of χ, this characteristic becomes more marked. However, market total shares of OD and OP firms are similar to those obtained in the other sets, confirming that parameter χ only influences the distribution of customers among the firms indipendently of their type. Therefore, the presence of χ in the ***Base Run*** has perfectly accomplished the purpose for which it was introduced keeping all the firms active in the market, without ousting many of them from the beginning.

Table 3.5: *Number of customers in the firms in the market for sets 2, and 2.1.*

	Set 2		*Set 2.1*	
Firm	OP customers			
	at $t = t_{entry}$	at $t = T$	at $t = t_{entry}$	at $t = T$
1	63	1,693	3,505	3,133
2	11,971	15,857	20,224	5,279
3	7,941	4,759	0	795
4	2,649	5,787	15,860	14,882
5	17,376	1,139	411	1,715
	Total number of OP customers			
	40,000	29,235	40,000	25,804
	OD customers			
	at $t = t_{entry}$	at $t = T$	at $t = t_{entry}$	at $t = T$
1	507	1,900	545	9,030
2	1,953	7,708	2,991	2,454
3	416	129	710	4
4	381	127	4	0
5	630	861	15	29
6	321	503	2,109	3,154
	Total number of OD customers			
	4,208	11,228	6,374	14,671

The results of *set 3*, obtained halving the standard deviation in the definition of coefficients a_j, do not show any variation in the purchase choices of the customers (see the Table 3.7 and the figure 3.6). This because the definition of their normal distributions both in this simulation set and in the ***Base Run*** does not privilege any type of firm. Indeed, the purchase choices of customers match the real ones. However, the purchase choices can be heavily influenced by the setting of these coefficients. For instance, if we define them as belonging to

Table 3.6: *Number of customers in the different firms in the market for sets 2.2, and 2.3.*

	Set 2.2		*Set 2.3*	
Firm	OP customers			
	at $t = t_{entry}$	at $t = T$	at $t = t_{entry}$	at $t = T$
1	2,730	3,288	2	2,976
2	19,449	9,344	0	602
3	13	5,181	19,871	16,010
4	17,520	1,251	0	3,853
5	288	4,235	20,127	2,194
	Total number			
	40,000	23,299	40,000	25,635
	OP customers			
	at $t = t_{entry}$	at $t = T$	at $t = t_{entry}$	at $t = T$
1	216	917	4,166	13,360
2	3,637	12,064	18	1,011
3	219	4	13	0
4	72	38	19	0
5	170	293	218	465
6	1,118	3,849	3	0
	Total number			
	5,432	17,165	4,437	14,836

the normal distribution $Norm(0.1, 0.01)$, the purchase choices are directed towards the least expensive products, because most coefficients are positive. Therefore, OP products, which have a higher price than OD products, entail a greater reduction of the utility function values, than their competitors.
On the contrary, if we define them as belonging to the normal distribution $Norm(-0.1, 0.01)$, the purchase choices are directed towards the most expensive products, because most coefficients become negative.
The results of *set 4* are reported in Table 3.7, and in figure 3.5. Comparing these results with the ***Base Run*** results, we find no significant difference. Neglecting the network effects does not entail any variation in the distribution of the customers. Therefore, the definition chosen in the ***Base Run*** is reasonable, because it does not privilige any type of product.

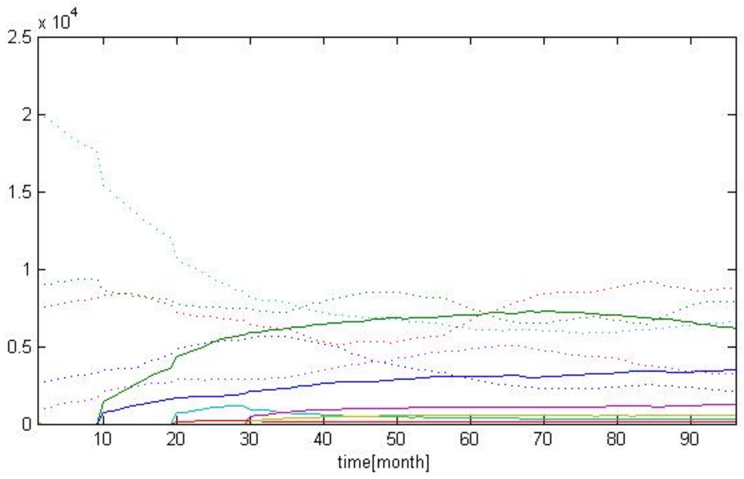


Figure 3.5: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 4*).

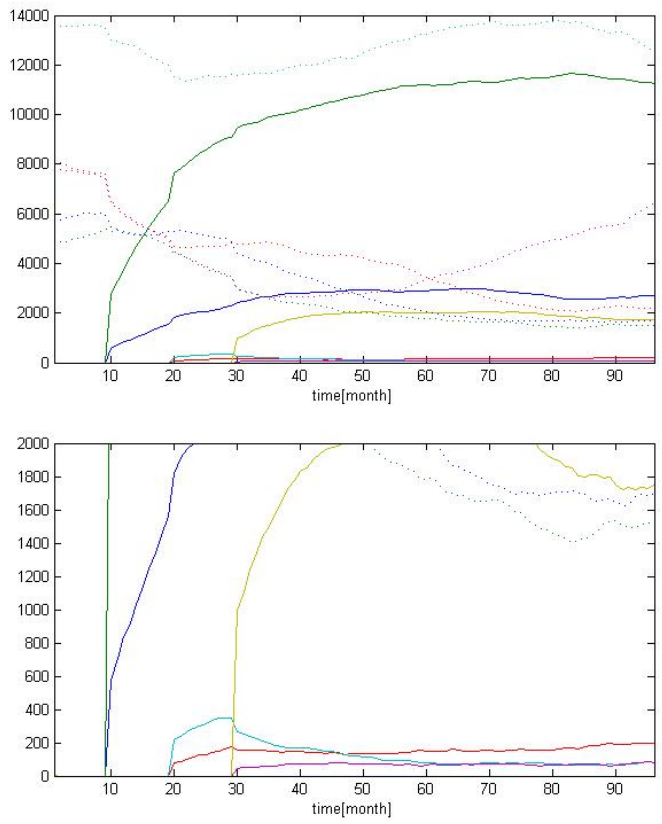


Figure 3.6: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 3*). The figure below expands y axis to highlight the customer distributions with smaller values.

Table 3.7: *Number of customers of the two type of firms for sets 2.i, 3, 4, 5.i, 6, 7, and Set 8.*

Set	OP users at		OD users at	
	$t = t_entry$	$t = T$	$t = t_entry$	$t = T$
2	40,000	29,235	4,208	11,228
2.1	40,000	25,804	6,374	14,671
2.2	40,000	23,299	5,432	17,165
2.3	40,000	25,635	4,437	14,836
3	40,000	24,444	5,487	16,034
4	40,000	28,616	4,218	11,873
5	40,000	22,646	8,547	17,847
5.1	40,000	22,754	8,547	17,743
5.2	40,000	22,183	7,073	18,292
6	40,000	24,346	6,231	16,134
6.1	40,000	30,395	4,325	10,071

The results shown in figures 3.7 - 3.9, concerning sets *5.i*, confirm what has been said for *sets 2.i*. The distributions of customers among the firms depend on the calibration of parameter χ, but they do not depend on parameter ϵ.

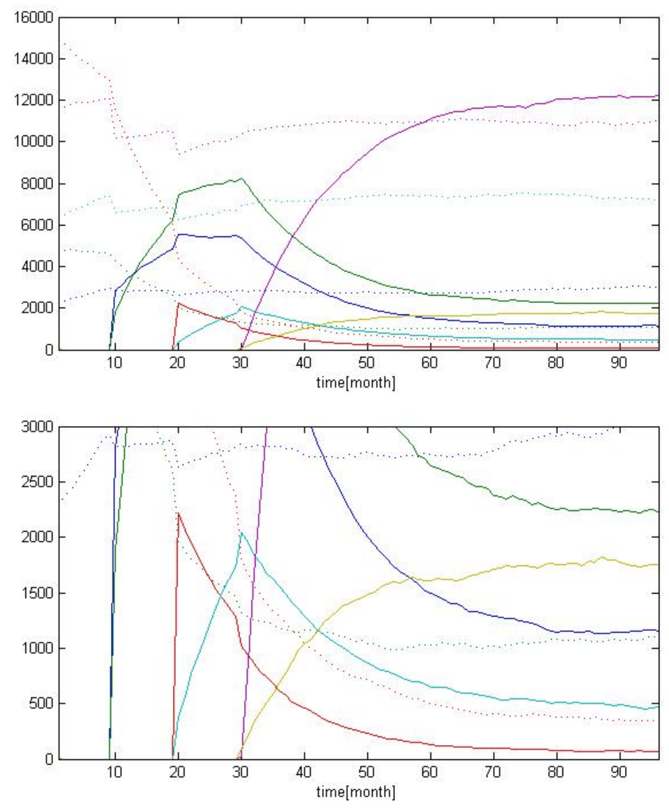


Figure 3.7: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 5*). The figure below expands y axis to highlight the customer distributions with smaller values.

In the followings, we will focus on the sensitivity of the model to term, η. This term models the customer feeling toward new Cloud technology. It models how customers evaluate the benefits and the risks associated to SaaS adoption. For this investigation, we run the *Set 6*, which repeats the ***Base Run*** neglecting parameter η. The results, illustrated in Table 3.7 and in figure 3.10, demostrate that this parameter does not influence the purchase choices of customers. The purchase choice for OD products persists also without the contribution of this parameter, which by definition increases the value of the utility function of OD products. To better investigate this issue the ***Base Run*** was repeated another time, *set 6.1*, neglecting both χ and η.

Comparing the results of *sets 2.3* and *6.i* (figures 3.10 and 3.11), we note that parameter η does not influence the market shares gained by the firms, as parameter χ does.

To conclude, we highlight that even if the results obviously depend on the chosen parameters' values, the performed sensitivity analysis demonstrated that the chosen parameters' values for the ***Base Run*** can be considered broadly correct, because it does not force the purchase choices of customers toward a specific firm or toward a specific type of firm. Indeed, the purchase choices of customers match the real ones.

We observed that the model is very sensitive to the variations of parameter χ,while is much less sensitive to the values chosen for all the other parameters.

Sensitivity analysis of the model to its initial parameters.

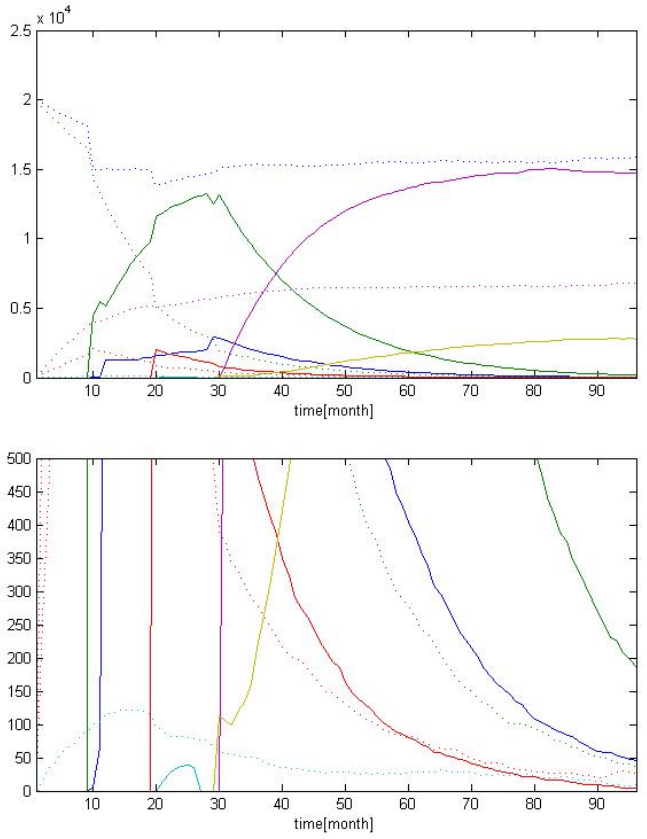


Figure 3.8: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 5.1*). The figure below expands y axis to highlight the customer distributions with smaller values.

In this section, we present further simulation aimed to investigate if also the simulation period, the firms and customers number, and the number of customers drawn at every time for re- evaluating their purchase choices, influence the distributions of customers among the firms. To this aim, we performed two simulation sets. In the first set, called *set 7*, we studied a market formed by 40,000 customers, 10 OP firms, and 9 OD firms, and run it for a simulation period equal to 156. In addition, we decreased the number of customers drawn at every time instant for re- evaluating their purchase choice. We set this number to $N_C/24$. The second set, called *set 8*, differs from the first only in the number of customers set equal to 20,000. All the other parameters not cited were set as in ***Base Run***.
These two sets were run twice: the first time (*set 7.1* and *set 8.1*) neglecting the random noise χ, the second time, *set 7.2* and *set 8.2*, neglecting both χ and η.
Finally, *sets 7.1* and *7.2* were run again (*sets 7.1.1* and *7.2.1*), decreasing the number of customers drawn at every time instant for re- evaluating their purchase choices. We set this number equal to $N_C/12$.

The results obtained with sets *7* and *8* are very similar each other, and for this reason, we report only the results of *sets 7.i.* In Table 3.8, we report the market shares got by the firms at $t = t_{entry}$ and at $t = T$. In Figures 3.12 and 3.13, we report the time trends of the distributions of customers among the firms. Comparing the results of *set 7* with those of the ***Base Run***, we

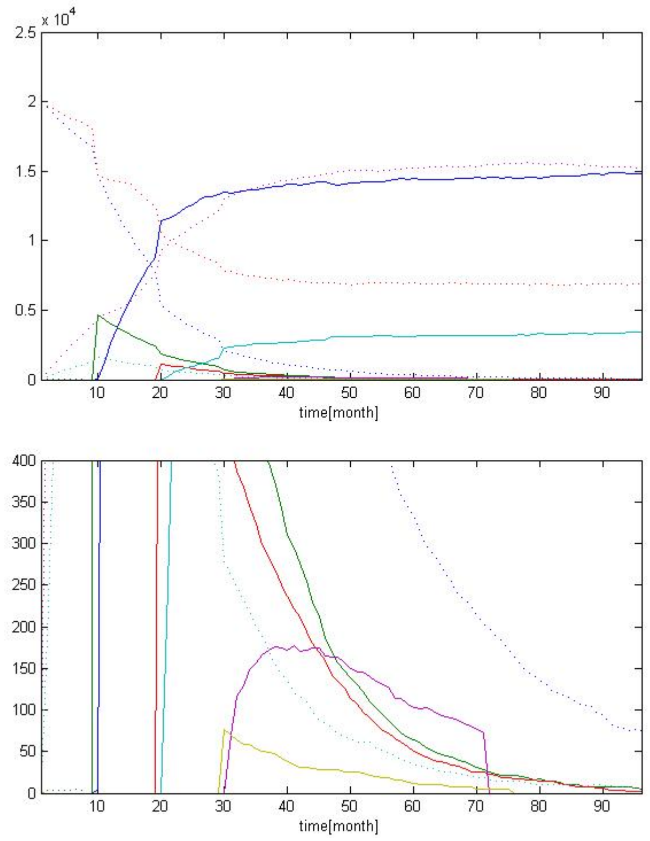


Figure 3.9: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 5.2*). The figure below expands y axis to highlight the customer distributions with smaller values.

do not observe significant differences.

On the contrary, the results of *sets 7.i* show that many firms are ousted from the market, and only eight or nine firms are able to survive, despite the total market shares got by the two type of firms do not vary with respect to the other sets. The distribution of customers among the firms is also in this case heavily influenced by the random noise χ, and not by parameter η.

For major clarity, in the Table 3.10, we report all the name of the simulation sets done, with a brief explanation of their main characteristics. In the Appendix A.1, we report in detail the data concerning the distribution of customers among the firms for all the simulations run and the figures of the customer distributions among the firms do not reported in this chapter.

3.2.3 Monte Carlo Analysis.

In the previous sections, we reported a sensitivity analysis to justify the chosen parameter values for the ***Base Run***. To assess the robustness of our model, we report a Monte Carlo analsysis performed. We repeated several simulation sets with the same initial conditions, but with different seeds of the random number generator.
For computational reasons and given that no significant difference in the outputs of the model has been emphasized with respect to the total number of customers and firms, and

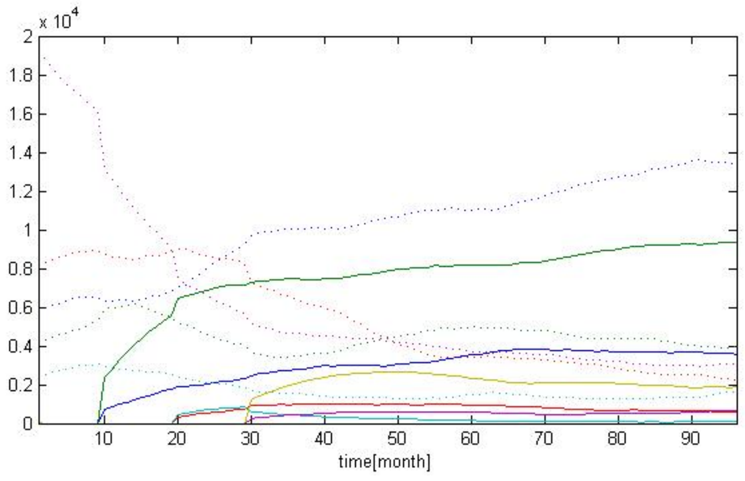


Figure 3.10: Customer distributions among the OP (dotted lines) and SaaS (solid lines) firms (*Set 6*).

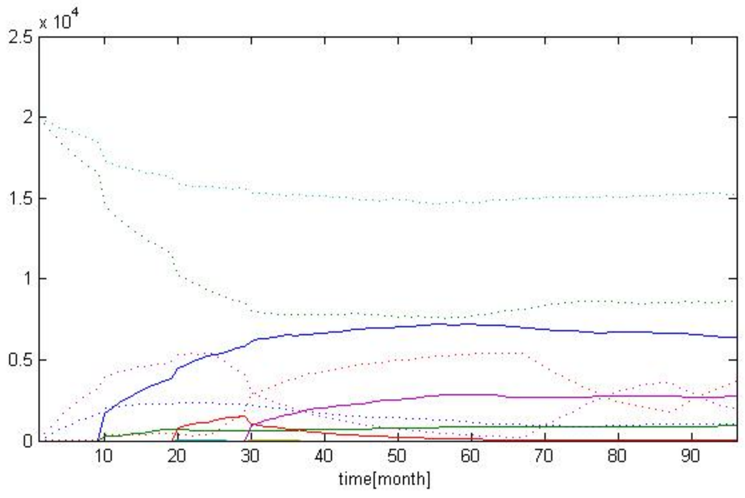


Figure 3.11: Customer distributions among the OP (dotted lines) and SaaS (solid lines) firms (*Set 6.1*).

with respect to the number of customers drawn to re-evaluate their purchase choices, the Monte Carlo analysis was performed running a modified version of the ***Base Run***, called *Base Run'*.
In this set we assumed a total number of customer equal to 20,000, a number of OP firms equal to 3, a number of OD firms equal to 4, and finally a number of customers drawn to re-evaluate their purchase choices equal to $N_C/24$, being all these numbers smaller than those of the ***Base Run***.
The *Base Run'* was run five times:

- the first time neglecting χ (*Base Run'.1*);
- the second time neglecting ϵ (*Base Run'.2*);
- the third time neglecting η (*Base Run'.3*);

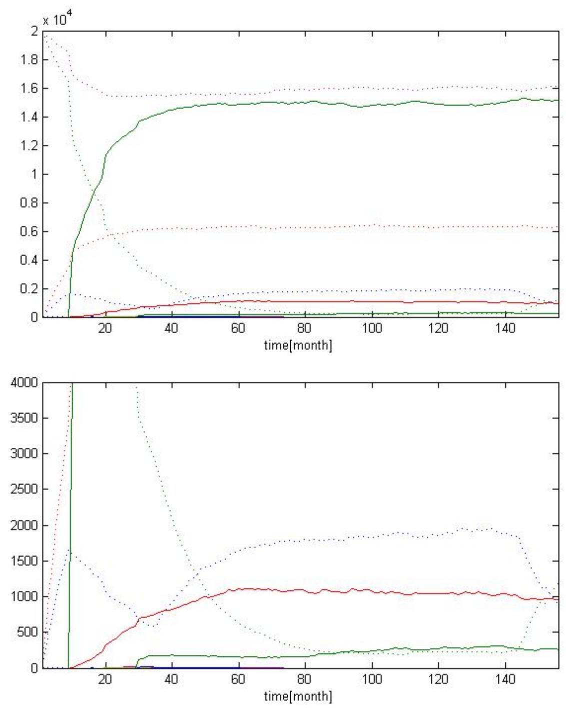


Figure 3.12: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 7.1.1*). The figure below expands y axis to highlight the customer distributions with smaller values.

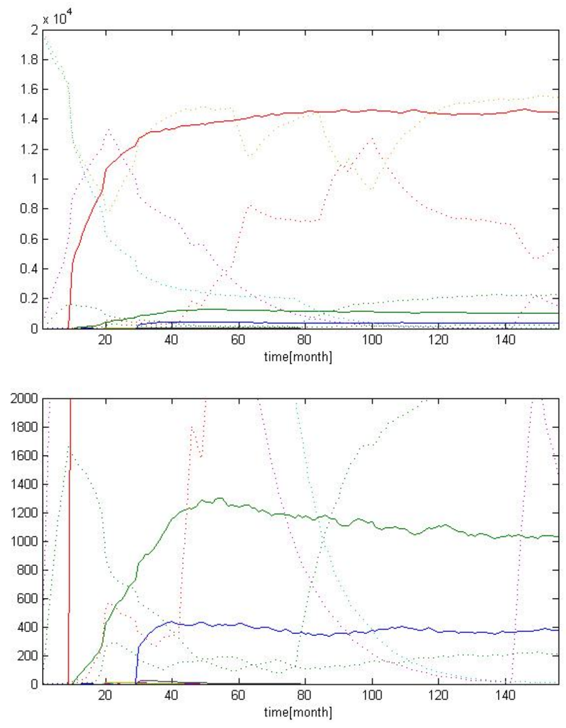


Figure 3.13: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 7.2.1*). The figure below expand y axis to highlight the customer distributions with smaller values.

Table 3.8: *Number of customers in the different firms in the market for the Sets 7.i.*

	Set 7		*Set 7.1*		*Set 7.2*	
Firm	OP customers at					
	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$
1	1096	400	1	10,015	0 ($t = 1$)	0 ($t_{exit} = 8$)
2	133	384	0 ($t = 1$)	0 ($t_{exit} = 8$)	52	15,715
3	283	3033	1 ($t = 1$)	0 ($t_{exit} = 8$)	0	568
4	292	350	20,159	12,298	0 ($t = 1$)	0 ($t_{exit} = 8$)
5	188	407	0 ($t = 1$)	0 ($t_{exit} = 8$)	19,911	6,281
6	416	385	19,839	2,023	0 ($t = 1$)	0 ($t_{exit} = 8$)
7	10479	9848	0 ($t = 1$)	0 ($t_{exit} = 8$)	20,036	4,957
8	274	407	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
9	18605	4649	0	220	1 ($t = 1$)	0 ($t_{exit} = 8$)
10	8234	8388	0 ($t = 1$)	0 ($t_{exit} = 8$)	(t_{exit} = 8)0 ($t = 1$)	0
	Total OP clients at					
	40000	28251	40,000	24,556	40,000	27,521
OD customers at						
1	1514	4469	12	977	2	330
2	855	2068	4,369	14,974	2,666	11,971
3	622	1862	0 ($t = 10$)	0 ($t_{exit} = 17$)	614	3
4	310	1252	12 ($t = 20$)	0 (t_{exit} = 121)	20 ($t = 20$)	0 (t_{exit} = 156
5	244	157	59 ($t = 20$)	0 ($t_{exit}l$ = 139)	706	3
6	456	1752	0 ($t = 20$)	0 ($t_{exit} = 27$)	0 ($t = 20$)	0($t_{exit} = 27$
7	86	188	11 ($t = 30$)	0 ($t_{exit} = 75$)	526	949
8	76	126	0 ($t = 30$)	0 ($t_{exit} = 37$)	0 ($t = 30$)	0($t_{exit} = 37$
9	350	651	312	283	17 ($t = 30$)	0($t_{exit} = 76$
Total OD customers at						
	4513	12525	4,775	16,234	4,551	13,256

Table 3.9: *Number of customers of the two type of firms in the market for Set 7.i, and Set 8.i.*

Set	OP users at		OD users at	
	$t = t_entry$	$t = T$	$t = t_entry$	$t = T$
7	40,000	28,251	4,513	12,525
7.1	40,000	25,816	4,059	14,932
7.2	40,000	26,904	4,055	13,912
8	20,000	12,687	2,858	7,783
8.1	20,000	12,993	2,196	7,488
8.2	20,000	14,874	1,979	5,562

We run 20 simulations for each above reported of the sets. For each Monte Carlo run and for each firm we stored the customer number at the entry time and at the exit time; in addition, for each Monte Carlo run we computed the number of surviving firms. In tables 3.11 - 3.12 we show the results of the Monte Carlo analysis. In particular, we report the 25th, 50th and 75th percentiles of the number of customers for the survived firms and for all Monte Carlo sets.

The Monte Carlo analysis confirmed all the considerations made for the sensitivity analysis.
Indeed, by making a comparison among the percentiles of the number of customers related to *Base Run'* with those of the *Base Run'.1*, *Base Run'.2*, and *Base Run'.3* , we can confirm the considerations made about the parameters η, χ and ϵ. The model is not sensitive to the

Table 3.10: *Sets of Simulation.*

Name Set	Definition
Set 1 or ***Base Run***	The values of parameter are defined as in the Tables 3.1, 3.2 and 3.3. In particular T=96, the number of customers drawn for evaluate the products in the market is equal to $N_C/12$, $\chi \in Norm(0,0.1)$, and the parameters a e b in the network effect are equal to 0.2 and 0.7, respectively.
Set 2	It repeats the ***Base Run*** and defines χ as Norm(0,0.05).
Set 2.1	It repeats the ***Base Run*** and defines χ as Norm(0,0.01).
Set 2.2	It repeats the ***Base Run*** and defines χ as Norm(0,0.0001).
Set 2.3	It repeats the ***Base Run*** and defines χ equal to 0.
Set 3	It repeats the ***Base Run*** and defines a_j as Norm(0,0.05).
Set 4	It repeats the ***Base Run*** and neglects the network effects.
Set 5	It repeats the ***Base Run*** and sets the parameters a and b in the network effects ϵ equal to 0.1 and 0.6, respectively.
Set 5.1	It repeats the *Set 5* and neglects both the network effects ϵ, and χ.
Set 5.2	It repeats the *Set 5*, takes the network effects ϵ into account, and neglects the parameter χ.
Set 6	It repeats the ***Base Run*** neglecting the parameter η.
Set 6.1	It repeats the ***Base Run*** neglecting both the parameter η and χ.
Set 7	It repeats the ***Base Run***, studies a market formed by 40,000 customers, 10 OP firms, and 9 OD firms, and runs it for a simulation period equal to 156. Further, it decreases the number of customers drawn at every time instant for re- evaluating their purchase choice, this number is set to $N_C/24$
Set 8	It differs from the *set 7* only for the number of customers equal to 20,000.
Set 7.1, 8.1	They repeat the *set 7* and *8* neglecting the random noise χ.
Set 7.1.1	It repeats the set 7.1 and re-increases the number of customers drawn at every time instant for re- evaluating their purchase choice to $N_C/12$
Set 7.2.1	It repeats the set 7.2 and re-increases the number of customers drawn at every time instant for re- evaluating their purchase choice to $N_C/12$
Set 7.2, 8.2	They repeat the sets *set 7* and *8* neglecting both χ and η

Table 3.11: *Monte Carlo analysis: Percentiles of the number of customers and total average number of survivor firms for Base Run', and Base Run'.1.*

		Base Run'		*Base Run'.1*	
Typology of firm	Percentiles	t_{entry}	at $t = T$	t_{entry}	at $t = T$
OP	$P_{0.25}$	5,117	3,062	7	2,624
OP	$P_{0.50}$	6,944	3,820	9,971	3,384
OP	$P_{0.75}$	8,659	4,952	10,024	6,316
OD	$P_{0.25}$	400	987	135	474
OD	$P_{0.50}$	537	1,667	606	1,475
OD	$P_{0.75}$	843	2,871	1,300	4,087
total average number of survivor firms		–	7	–	6

values chosen for the parameters η and ϵ, but it is very sensitive to the parameter χ. Indeed, *Base Run'.1* is the only set in which the Monte Carlo analysis resulted in the exit of some firms from the market, computing a total average number of survived firms equal to 6.

3.3 Summary.

This chapter presents a simulation-based approach to analyze the software market, and in particular the CRM software market, a complex dynamic system.

Table 3.12: *Monte Carlo analysis: Percentiles of the number of customers and total average number of survivor firms for Base Run', and Base Run'.1.*

		Base Run'.2		*Base Run'.3*	
Typology of firm	Percentiles	t_{entry}	at $t = T$	t_{entry}	at $t = T$
OP	$P_{0.25}$	5,293	2,887	4,891	2,520
OP	$P_{0.50}$	6,839	3,525	7,211	3,221
OP	$P_{0.75}$	8,879	5,624	9,027	6,064
OD	$P_{0.25}$	315	659	327	662
OD	$P_{0.50}$	539	1,536	515	1,748
OD	$P_{0.75}$	768	3,041	840	3,114
total average number of survivor firms		–	7	–	7

We proposed a model to study the competition among OP firms, which enter the market at initial time, and OD firms, which enter the market at next time instants.
Given the limited quality of the data available, it makes sense to analyse the business trends in an ideal CRM market reproduced by us through the ***Base Run***, where we adopted parameter values which can be considered broadly correct. To justify the chosen values for the ***Base Run*** a detailed sensitivity analysis was performed, studying what happens while varying the values of the different parameters. In addition, the robustness of the proposed model was evaluated with a Monte Carlo analysis.
We studied the CRM market with a smaller number of firms and customers than the real one due to computational reasons, and in which the CRM solutions are directed to a midmarket formed from enterprises customers with 100-1000 employees according to [62].
The simulation results, of course, depend on the values of the structural parameters chosen. It is worth to underline, that the chosen values for the parameter χ are those that influence a lot the distribution of customers among the firms of the same type. This because, they have been chosen with the aim not to oust the most firms from the beginning, allowing us to study a no too small market. Indeed, by turning on or turning off, increasing and decreasing the values of the different parameters, we observed that the model is very sensitive to the variations of the parameter χ, while is less sensitive to the values chosen for all the other parameters.
The setting of the other parameters such as a_j, network effects, σ, simulation period, number of firms and customers does not seem to heavily influence the purchase choices of customers, and consequently the distribution of customers among the firms. Indeed, this setting does not seem to force the purchase choices of the customers towards a specific firm, or towards a specific typology of firm. The customers buy the products according to their preferences as in reality. They privilege some products with respect to others, and consequently, they make the market shares of some firms very big.
Although there is relatively little empirical evidence available about the strengths of the various mechanisms on hand, the small market reproduced matchs the real one. In fact, the results in the ***Base Run***, as it happens in the real software market, emphasize that only few firms are able to conquer big market shares, while the remaining firms stay in the market with much smaller market shares.
Moreover, also the features of the products modelled match the real ones. Indeed, the results highlight that the values of the OP CRM solution quality are bigger than that of the OD CRM solution quality, in agreement with the fact that OP CRM solutions are more customized than

OD CRM solutions [47]. In addition, the prices of OP CRM vary between about 300 and 1,000 for the primary products, and between about 80 and 260 for secondary products; instead, the prices of OD CRM vary between about 180 and 350.

As has been stated before, the lack of reliable information with respect to the parameters and the model structure, limits qualitative conclusions, even though we feel that the parameter value and the model structure we have chosen are broadly correct, and allowed us to obtain quite a realistic reproduction.
Let us underline that the results reported aim only to show the potentialities of our model. Surely, the most significant results can be obtained calibrating the model with real data.

Chapter 4

On Demand and On-Demand FLOSS CRM Software Market Simulation Model.

Starting from the model presented in chapter 3, we propose in this chapter a modified version for analyzing and studying a specific segment of the CRM software market. In particular, the model analyzes the competition among On-Demand (OD) firms offering Customer Relationship Management (CRM) products with and without source code availability.

We studied two of the three generations of CRM systems taken into examination in a work by SugarCRM [64]. In this work the authors identified and compared the total cost of ownership among three generations of CRM systems.

- The *first generation of CRM systems* (eg. Siebel) is formed by closed proprietary systems. They are expensive to purchase, customize and administer and difficult to install. Only large organizations could take advantage of these CRM.
- The *second generation of CRM systems* (eg. salesforce.com) includes the CRM solution sold as a subscription-based service and hosted by the vendor. These solutions require a small upfront capital commitment and no infrastructure burden as the solutions of the first generation. They become popular among small businesses, while the larger organizations found extremely difficult to make them to comply with their complex requirements. This due to the proprietary architecture that limits flexibility, especially regarding security, scalability, and customization.
- Finally, the *third generation of CRM systems* (eg. Sugar CRM Professional) blends the benefits of the two previous generations customizability, control, security, low or no initial capital expenditures, full database and application control, and no vendor lock-in. In this generation, the providers offer to the customers the source code of their products.

In a traditional first or second generation of CRM software product, the contractual power shifts from the customer that chooses the CRM to the software provider. The customer is wholly dependent from the provider, that releases only binary code. So, the customer looses the control of implementation decisions. The provider has the source code and does not

Table 4.1: *Pricing Options for CRM products [64].*

Product	Pricing per User	Total Yearly Price per User	Source Code Availability
Sugar Professional	$239/user/year	$239	yes
salesforce.com	$75/user/month	$1,500/user	no
Siebel CRM	Priced for user for perpetual licensing: $2,000	$2,500	no

give to the customer the possibility of studying, analysing and/or modifying the code for customizing and adopting it to his own needs. For example, the software provider decides if and when to make new functionalities available, or if and when to fix bugs.
On the contrary in the case of open source CRM solutions (third generation solutions), the user is relatively independent from the provider. Consequently, if he wishes to fix a bug immediatly, he can make the fix itself, he can obtain the fix from the open source development community, or he can wait for the fix to be included in a future release.
The different characteristics and implications of the three CRM solutions, highlight the need of making a careful analysis of the costs and benefits of the different products in the market before choosing a software solution. In [64] the authors show how the adoption of a CRM provided with the source code yield to a saving of about 88% with respect to the adoption of a CRM provided without the source code. However, it is worth noting that price is only one factor of the Total Cost of Ownership, and the software third generation solutions offer a lower annual cost, thanks to the several benefits associated to the use of open source software. Total Cost of Ownership is formed in fact from many expenditure items [41]: cost of up-front evaluation study, cost of up-front proof of concept implementation, cost of vendor lock-ins, acquisition cost of software, cost of customisation for business needs, cost of integration to current platform, cost of migration (data and users), cost of training, cost of support services, cost of maintenance and upgrades, exit costs (in relation to hardware and software), and exit costs (in relation to changeover and re-training).

These expenditure items could be reduced by adopting open source software solutions. Among others, we can list the following factors [41]: reduced vendor lock-in, ability to experiment or innovate, value for money, easier access to knowledge and skills, better business agility, support for incremental development of solutions, ability to build and work with a peer community to re-use and share code, ability to work with local/SME service providers, access to a wider choice of support service providers, ability to work with sector peers on common areas of interest, full adoption of open standards, access to the code in case it is needed, ability to modify the code (e.g. for customization and solving critical defects), and ability to change support service providers.

In Table 4.1 we report the pricing options for CRM products reported in the work by SugarCRM [64]; the same values were reported by Chris Bulcholtz in [8]. Besides Sugar CRM, there are many other products provided with code availability. David Hakala cited the following [14]: CentricCRM Hypergate, Compiere Inc., Vtiger CRM, CentraView Inc., XRMS CRM, Cream CRM, and Tustena CRM, and underline that "although many companies opt for major CRM offerings such as Salesforce and Oracle, open-source solutions are proving to be popular among businesses with limited costs and unique needs".

In addition to the work [64], also others authors in the works [9], [53], [67] underlined that open source CRM solutions are less expensive, more easily modified by internal IT staff, or by external developers, and can be a good choice for having a more flexible development environment.
Finally, let us underline that, in 2012 the leaders of the Midmarket Suite CRM market were Microsoft, NetSuite, Oracle and SugarCRM; instead, for Open Source CRM they were: Adempiere, Concursive, Consona, vTiger, ([26]). The winner for the two categories are Salesforce.com and SugarCRM respectively.

4.1 The model proposed: firms and customers.

In this chapter we apply a modified version of the model described in the previous chapter, in order to analyse a software market in which enterprise customers, Closed Source CRM System vendors (CS), and Open Source CRM System vendors (OS) interact. We do not report the whole model structure but only the modifications applied to the model in order to fit it to the study of the new type of vendors, Open Source CRM System vendors (OS).

Consequently, the investment and pricing policies and the features of the products are modelled using the equations associated to the OD firms presented in Chapter 3, and in the following, we report only the definition of the quality for the OS product. The definition of the quality of this products is given by:

$$\left(\frac{dQ_i}{dt}\right) = \alpha(t) * m(t) * \lambda_m(t) + h_i(t)N_i - \delta Q_i \tag{4.1}$$

where :

- α: is the involvement level of the open source user community;
- $m(t)$: is the size of the open source community at time t;
- λ_m: is the fraction of the whole community that contributes to the product development.

This equation, in addition to the contributions of the developers in the $i-th$ firm, takes the contributions of the open source community into account. This to model the fact that, customers who adopt open source solutions are independent from the provider, and can obtain the support of the open source community to fix bugs, improve the code or also customize their applications [64].

4.2 Base Run and other simulation set results.

In this chapter, we report the results of the implementation of the model which represents a vertical software market, in which compete On-Demand Suite CRM, and Open Source CRM [21].

We study a vertical market in which the CRM solutions are directed towards a midmarket formed from enterprise customers with 100-1000 employees [62].

The simulated market is an ideal market, in which ten firms and 40,000 enterprise customers enter the market at the initial time $t = 0$. The simulation spans over 96 months, the

simulator step being equal to one month. Every firm enters the market proposing its products and the customers choose the products to acquire.
We calibrated the model with the aspiration to obtain the price of the OS products equal to 88% of the price of the CS products, as in [64], and to have their orders of magnitude similar to the real ones.

Moreover, to obtain a market in which few firms are able to survive, we characterized the parameter χ through a normal distribution indicated as $Norm(0, 0.001)$, in which the first term is the average, and the second is the standard deviation.

In addition, in order to obtain a market in which also OS firms are able to survive, we defined the parameter η in the following way:

- for OS firms having more than 500 end users $\eta \in Norm(0.00005, 0.00001)$;
- for OS firms having less than 500 end users $\eta \in Norm(0.0001, 0.00005)$;
- for CS firms $\eta = 0$.

In Tables 4.2 and 4.3, all the parameters' values of the model are reported. Some of these values are taken from literature, and market analysis, others are set in order to obtain prices that match the real prices in the market.

At first, we planned three sets of simulations, identified with the name: *A*, *B* and *C*. Each of them had different values for the parameters associated to the contribution of the open source community to the development of the OS products. This was done to compare the quality of the OS and the CS products, and analyze how the quality of the OS products could increase in response to a growth of the involvement of the open source community in the development of the product.
Remember that, by quality of CRM product we mean all the software features which contribute to give value to it, and which are *Functionality*, *Reliability*, *Efficiency*, *Usability*, *Maintainability*, and finally *Portability*. Of course, these features are present in both typologies of product, but in the OS case they can be improved taking advantage of all the benefits associated to OS software [19]. For instance, we can increase the product quality, using the contribution of the open source community, or exploiting the abilities of different suppliers. In fact, Open Source Code means independence from the individual providers, and consequently maintaining a greater decision-making autonomy in scheduling the updates with respect to closed source code. Adopting open source products, thanks to the access to the code, means to give to the customer's IT team the opportunity of growth studying and modifying the code. This allows customers to customize the software according to the real needs. Moreover, the adherence to standards and the availability of source code provide complete access to the user's data, and allows customers to safeguard the investments made in terms of hardware and software without being exposed to the strategic choices of the suppliers.
Back to our simulation, in set *A*, we run the model, neglecting the contribution of the open source community, while in the other two sets, *B* and *C*, we took it into account. In particular in the latter two sets, we analyzed how the features of the products, and the time trend of the market vary while increasing the OS community contribution. Indeed, in set *C*, we assume a higher value of this contribution than in set *B*. In set *B* the parameter λ_m varies in the range $[0.002, 0.005]$, and in set *C* it varies in the range $[0.02, 0.05]$.

Table 4.2: *The table reports the values of the parameters of On-demand vendor equations.*

Parameter	Description	Value
N	Number of developer.	Normal variable with average $\mu = 165$ and standard deviation $\sigma = 11$.
A_i	Initial capital available to each company	80,000.00 per employee
β_i	Fraction of the initial capital invested by $i-th$ firm.	Normal variable with average $\mu = 0.9$ and standard deviation $\sigma = 0.03$.
Y_i	Fraction of capital invested for hardware expenses.	Normal variable with average $\mu = 0.45$ and standard deviation $\sigma = 0.02$.
r_i	Fraction of capital invested monthly.	Normal variable with average $\mu = 0.003$ and standard deviation $\sigma = 0.00033$.
ω	Fraction of capital invested monthly in hardware and network expenses.	Normal variable with average $\mu = 0.0004475$ and standard deviation $\sigma = 0.000025$.

To predict the trends of the CRM market in response to a shift of the purchase choices of customers towards OS firms, we run the *set B* also called the ***Base Run***. Indeed, in all the three sets we adopted parameter values according to the data come from the literature, market analysis and our experience. However, in the Base Run we choose for the parameter λ_m a value which can be considered quite a reasonable for our goal: studying the CRM market trend when the purchase choices of customers move towards the OS products.
Now, we analyse the results of the ***Base Run***. In the figures from 4.1 to 4.4, we show the time trends of the quality and the price, and the distributions of customers among the firms.
The former figure shows how the quality of OS products assumes higher values than that of CS products, confirming that the contribution of OS community can influence positively the quality of OS products.

For major clarity in Table 4.4 we report the quality values, at different time instants, for the products which show the highest values of the quality. Consequently, these products can be considered the best products in the market.

From the figures 4.2 and 4.3, it is possible to observe that the prices of the OS products

Table 4.3: *The table reports the values of the parameters of On-demand vendor equations.*

Parameter	Description	Value
δ	Quality depreciation rate:	0.03. (Haruvy, Sethi and Zhou [16])
π	Productivity of human capital accumulation process	0.025. (Ghosh, p.237 [52])
γ	Fracion of human and ICT capital in human capital accumulation process:	0.9. (Ghosh, p.237 [52])
u	Fraction of time spent on human capital formation h,:	0.1 (Ghosh, p.237 [52])

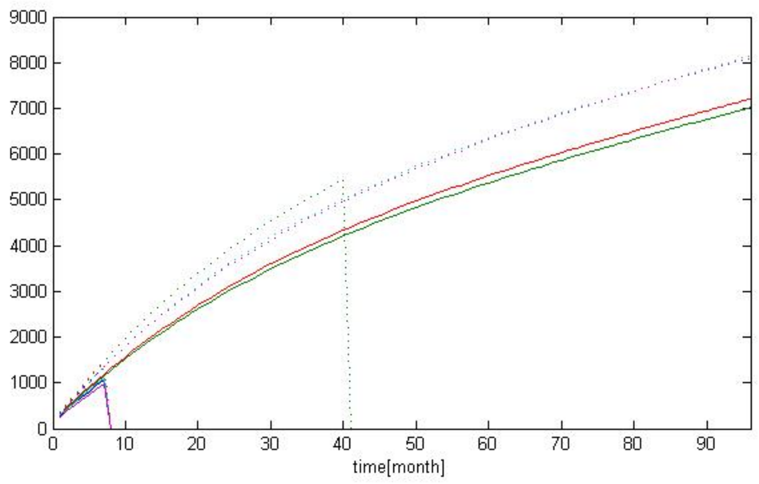


Figure 4.1: Quality of CS (solid lines) and OS (dotted lines) product (***Base Run***).

are about 88% of the prices of the CS products, and hence they match the real ones [64].
Finally, in the figure 4.4, the distributions of customers among the firms are reported. Only four firms out of ten survived in the market and the firm which got the biggest market share was an CS firm. For the survived OS firms a particular trend can be emphazised.
The survived OS firm which prevails in the market at first sees to reduce and then to re-increase its market share in favour of another OS firm staying in the market.
Note that, the market total share got from the two types of firms do not differ a lot from each other. Indeed, CS firms got 51% of the market, while OS firms got 59%.
In figures 4.5 and 4.6 we report the time trends of the distributions of customers among the firms for *sets A* and *C*. These distributions highlight that only four or five firms out of ten are not ousted from the market, and only two of them are able to survive in the market with big market shares. These two firms are one OS and the other CS.

Table 4.4: *Quality Values of best OS and CS products in the market.*

Base Run		
Time	*Best CS product quality.*	*Best OS product quality.*
$t = 10$	1,576	1,796
$t = 20$	2,693	3,087
$t = 30$	3,594	4,118
$t = 40$	4,340	4,971
$t = 50$	4,975	5,675
$t = 60$	5,531	6,318
$t = 70$	6,034	6,867
$t = 80$	6,502	7,366
$t = 90$	6,948	7,835

Moreover, we can observe in *set A* the particular trend seen in the ***Base Run***. OS firm which prevails in the market at first sees to reduce and then to re-increase its market share in favour of another OS firm staying in the market. Similar behaviour is present also in *set C*. The results of *sets A* and *C* are perfectly in accordance with the results which we wished to obtain. Indeed, in agreement with the values chosen for the parameter λ_m, in *set C* the results showed the highest values for OS product quality, and in *set A* the lowest values for OS product quality. In addition, in *set C* an OS firm got the biggest market share.

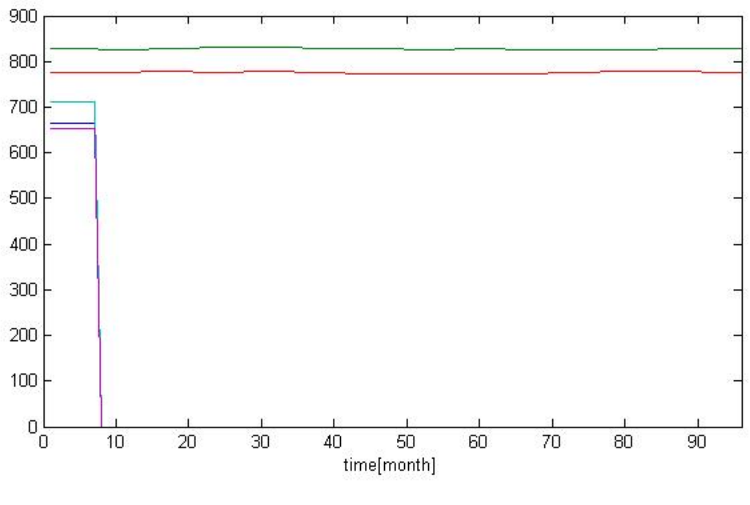


Figure 4.2: Price of CS products (***Base Run***).

4.2.1 Sensitivity Analysis.

In this section, we report a sensitivity analysis of the model to the coefficients a_j, the simulation period, the number of firms and customers, and the number of customers drawn to re-evaluate their purchace choice at each time, to justify the chosen parameter values for the

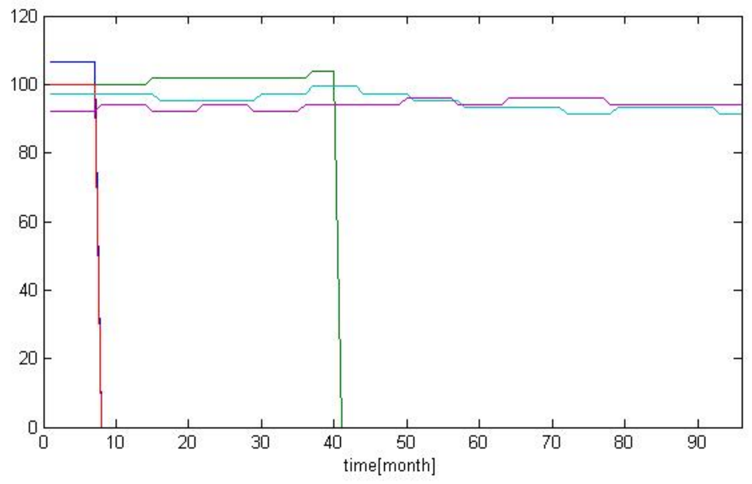


Figure 4.3: Price of OS products (***Base Run***).

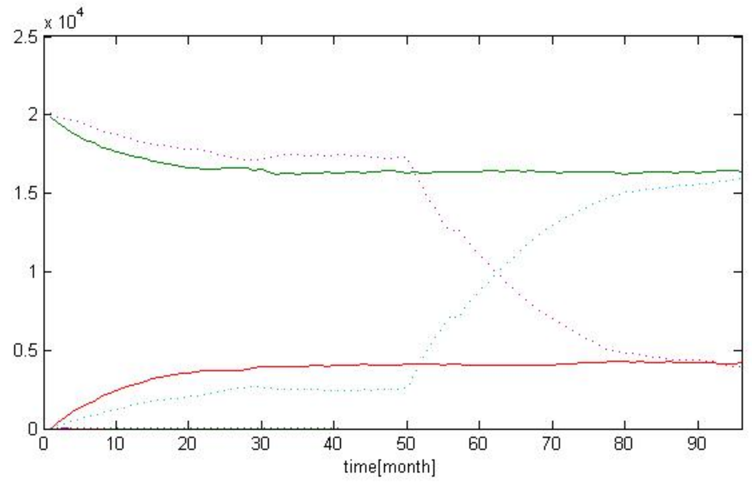


Figure 4.4: Customers of CS (solid lines) and OS (dotted lines) firms (***Base Run***).

Base Run.
In the three sets analysed in the previous section 4.2, all the coefficients a_j are characterized by a normal distribution with average 0 and standard deviation 0.001, and therefore, they direct the purchase choices of customers among the two types of firms without creating great differences between the market total shares of the two types of firms.
For investigating the influence of the coefficients a_j over the purchase choices of customers, we performed two further simulation sets, the *sets B.1* and *B.2*. In these sets, we calibrated the coefficients a_j in order to direct the purchase choices of the customers towards the OS products, which are less expensive and characterized by a higher quality than CS products, when there is a great contribution of the open source community to the development of the product.
Remember that, the coefficients a_j contribute to customize the purchase choices of the customers, weighting the different terms in the utility function.
In the *set B.1*, the coefficients a_Q, a_P and a_{Sc} acquire their values from two different normal distributions.

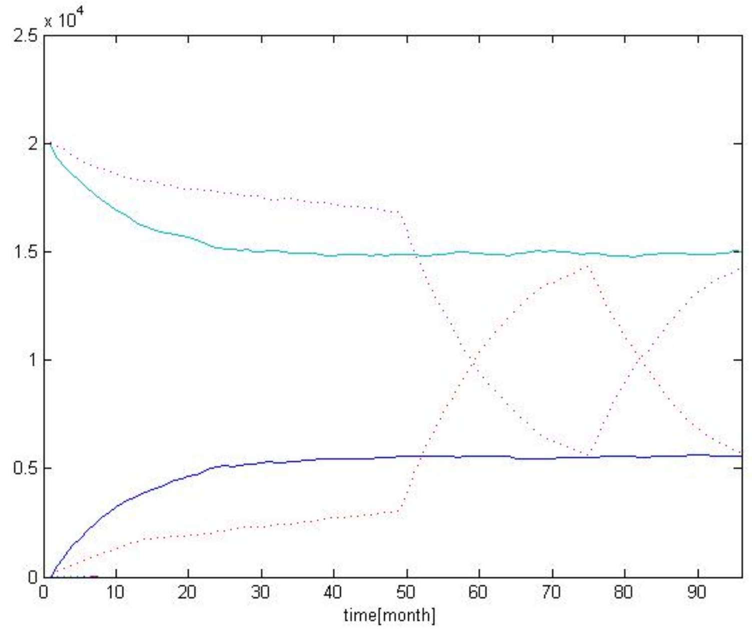


Figure 4.5: Clients of CS (solid lines) and OS (dotted lines) firms (*Set A*).

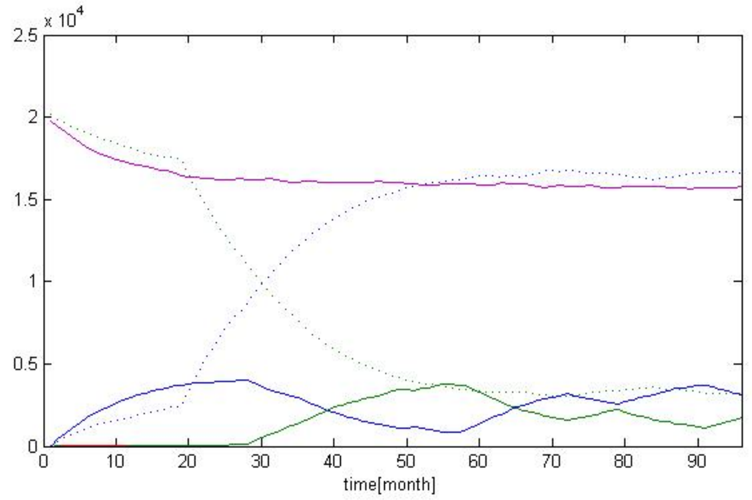


Figure 4.6: Clients of CS (solid lines) and OS (dotted lines) firms (*Set C*).

The first is a normal distribution with average 0.1 and standard deviation 0.01, and the second is a normal distribution with average 0 and deviation standard 0.001. In this way the coefficients which acquire their values from the first distribution contribute to increase the value of the utility function for the OS products, which are less expensive and characterized by a higher quality than CS products. While the coefficients which acquire their values from the second distribution contribute to distribute in uniform way the purchase choices of the customers between the two types of firms.
On the contrary, in the *set B.2*, the coefficients a_Q, a_P and a_{Sc} acquire their values only from the first normal distribution, directing more and more the purchase choices of customers towards the only OS products.

The results, in the figures 4.7 and 4.8, confirm the sensitivity of the model to the calibration of the coefficients a_j. Acting on them, we can privilege one of the two types of firms. The results in *set B.1*, highlight the success of the OS firms, which get almost one of third of the whole market. Instead, in *set B.2* only one OS firms is able to stay in the market and to get almost the whole market. In fact, all the other OS firms are ousted, and only one CS firm survives in the market with a market share very small equal to 25.

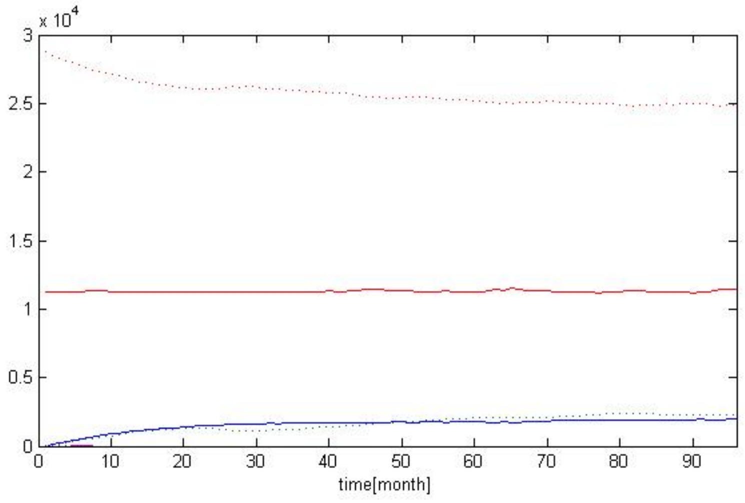


Figure 4.7: Clients of CS (solid lines) and OS (dotted lines) firms (*Set B.1*).

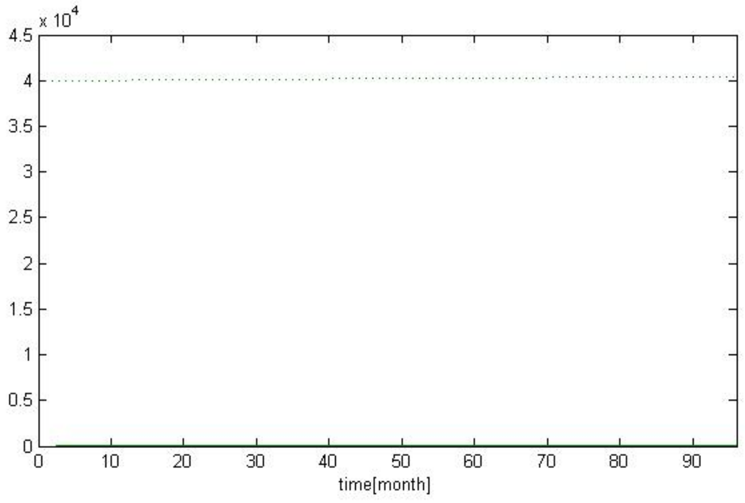


Figure 4.8: Clients of CS (solid lines) and OS (dotted lines) firms (*Set B.2.*)

Further, three simulation sets, called *set D*, *E*, and *F*, were run to investigate, if there are other parameters in addition to the parameters χ and η, which influence the distribution of customers among the firms. All the parameters of these three sets were similar to those of the *set B*, except for those cited below. In *set D* we set T equal to 156, and the number of customers N_C equal to 20,000. *Set E* is similar to *set D* except for the number of firms, which was set equal to 20. At the end, *set F* differs from *set E* for the number of customers drawn to re-evaluate their purchase choice. This number was set equal to $N_C/24$.

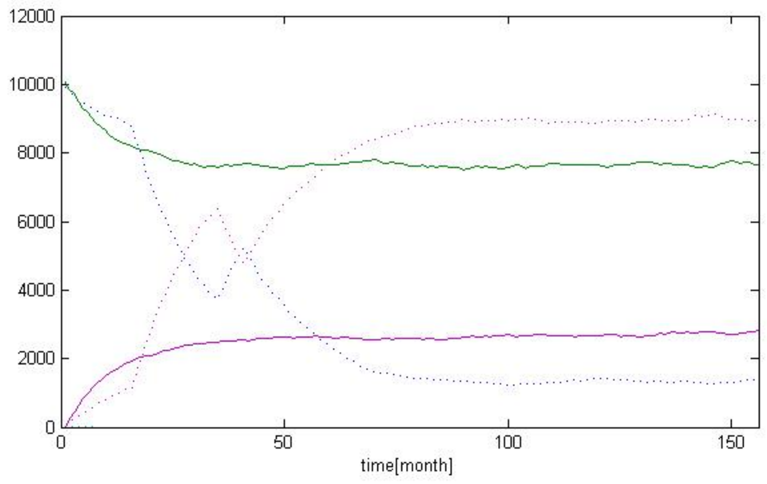


Figure 4.9: Clients of CS (solid lines) and OS (dotted lines) firms (*Set D*).

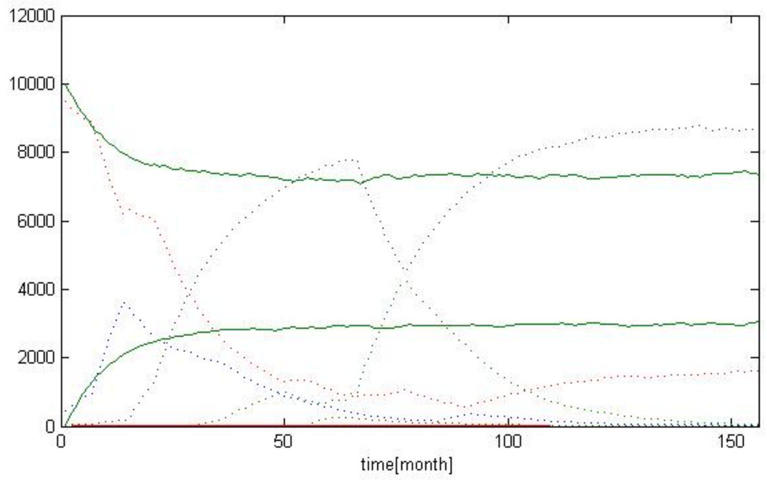


Figure 4.10: Clients of CS (solid lines) and OS (dotted lines) firms (*Set E*).

The figures 4.9, 4.10 and 4.11 show that, the distributions of customers among the firms are not sensitive to the simulation period, the number of firms and customers in the market, and the number of customers drawn to re-evaluate their purchase choice at each time. The customers distribute themselves among the two types of firms, as in *sets A*, *B* and *C*, in which the market share are equally distribute among the two types of firms.
In the light of the analysis done, we can conclude that all the parameters' values can influence the results of the model. But, the chosen values for the***Base Run*** can be considered broadly correct for our goal. The parameters which heavily influence the distributions of the customers between the firms are the parameter λ_m, the random noise χ and the parameter η and indeed, thanks to them we forced the purchase choices of customers toward OS products ans for this reason many simulations was run to choose their best calibration.. The first parameter introduces a random noise in the utility function, and allows us to distribute the purchase choices of customers in a more or less uniform way among the firms; the second increases the value of the utility function for OS products.

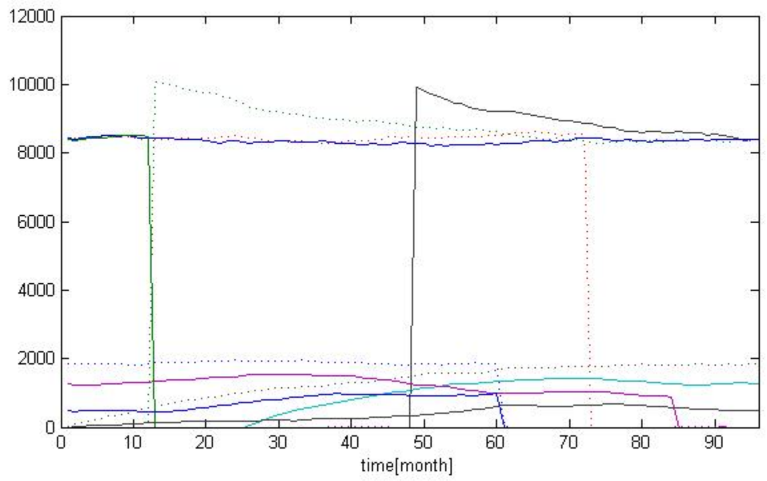


Figure 4.11: Clients of CS (solid lines) and OS (dotted lines) firms (*Set F*).

4.2.2 Monte Carlo Analysis

In the previous section, we reported results about some simulations and a sensitivity analysis performed in order to justify the chosen parameter values for the ***Base Run***. Now, to assess the robustness of our model, we presented the Monte Carlo analysis performed. We repeated several simulations with the same initial conditions, but different seeds of the random number generator. We executed 20 simulations for the sets B, D, E and F. In particular, for each Monte Carlo run and for each firm we stored the customer number at the entry time and at the exit time, in addition for each Monte Carlo run computed the number of firms survived. In the tables 4.5 and 4.6 we show the results of the Monte Carlo analysis, reporting the values of the percentiles of the number of customers for *sets B*, *D*, *E*, and *F*, obtained taking only the firms survived into account.

In particular, we report the 25th, 50th and 75th percentiles of the number of customers for all Monte Carlo runs, and for every type of firm.

Table 4.5: *Monte Carlo analysis: Percentiles of the number of customers and total average number of firms survived for sets B, and D.*

		Base Run		*Set D*	
Typology of firm	Percentiles	t_{entry}	at $t = T$	t_{entry}	at $t = T$
OS	$P_{0.25}$	12.75	2,190	8.25	1,176
OS	$P_{0.50}$	5,852	9,857	41	4,114
OS	$P_{0.75}$	20,177	17,596	10,168	9,003
CS	$P_{0.25}$	5	3,143	9,719	1,495
CS	$P_{0.50}$	19,571	5,560	9,778	2,598
CS	$P_{0.75}$	19,778	15,080	9,823	8,150
total average number of firms survived			4.65		4.55

The Monte Carlo analysis associated to the *Base Run* confirmed the distributions of customers among the firms reported in the previous section regarding the *Base Run*.

Table 4.6: *Monte Carlo analysis: Percentiles of the number of customers and total average number of firms survived for sets E, and F.*

		Set E		*Set F*	
Typology of firm	Percentiles	t_{entry}	at $t = T$	t_{entry}	at $t = T$
OS	$P_{0.25}$	7	1,069	5	803
OS	$P_{0.50}$	86	1,768	33	3,636
OS	$P_{0.75}$	10,004	8,736	9,999	8,685
CS	$P_{0.25}$	3	1,228	2,460	350
CS	$P_{0.50}$	9,060	2,543	9,916	2,625
CS	$P_{0.75}$	9,921	7,800	9,960	7,600
total average number of firms survived			5.1		5.25

In addition, the results of the Monte Carlo analysis concerning *sets D, E,* and *F* confirm that the model is not sensitive to the parameters T, N_C, N_F and $N_{C,drawn}$.
Note that, detailed data on all the simulations run are reported in the Tables from the A.6 to the 4.6, in Appendix A.2.

4.3 Summary.

In this chapter, a business model is proposed for analyzing and studying a specific segment of the software market. In particular, the model analyzes the competition among the OD firms offering Customer Relationship Management products, with and without the source code availability.
We would like to underline that the model could be easily modified to study other segments of market.
Moreover, the proposed model could be used as simulation tool by project managers to analyze how their investments or pricing mechanisms might facilitate or hinder the conquest of bigger market shares.
The results obtained highlight that the market can move towards the software solutions provided with the source code availability, and that this entails a positive influence on the whole software market. Indeed, the source code availability allows customers to safeguard the investments made in terms of hardware and software, without being exposed to the strategic choices of the supplier, maintaining a greater decision-making autonomy in the scheduling the updates than the closed source code.

The availability of source code and the widespread knowledge of the technologies used in the development of open source products, such as programming languages and communication techniques, guarantee the independence from individual providers, increasing the bargaining power of end users and encouraging the creation of a more competitive software market.

The systematic use of software solutions with the availability of source code leads the development of local skills, necessary to the support and customization of the software, helping local high-tech jobs. Source code availability means opportunity of formative growth and customization of the software according to the real needs. The adherence to the standards and the availability of sources provides complete access to data by user. This is not an inefficient model, but the overcoming of inefficient and dangerous monopolies, with a more modern and efficient service-oriented technological model. Customer relationship management products with source code availability, but also different software solutions with this feature, may gain bigger market shares in the near future.

Chapter 5

Comparison among the Results the Simulation Models.

5.1 Some considerations about the results in chapters 3 and 4

In this section, we briefly discuss about the results obtained applying the model proposed to the two different study cases, reported in the two previous chapters.
The base structure of the model applied is the almost same in the two cases, but the sensitivity of the model to some parameters is different. In particular, the sensitivity analysis performed has showed different sensitivity to the parameter η. Indeed, in the simulation of the competition among OD and OP vendors this parameter does not influence the distribution of customers among the firms; on the contrary, in the simulation of the competition among OD vendors, who distribute their product with and without the source code availability, the results emphasize a great sensitivity to this parameter.
Let us remember that, the parameter η contributes to increase the value of the utility function associated with OD products in the model presented in Chapter 3, and the value of the utility function associated with OS products in the model presented in Chapter 4.
This different sensitivity to the parameter η is surely linked to the different composition of the market.
Indeed, in the first model we have very different vendors among them. They are characterized from investment and pricing different mechanisms, and in general from business different policies, which entail the market positioning of products having very different features from each other.
Consequently, the values of these features (Q_i, P_i and ρ) play a very important role in the formation of the utility function values, contributing in a relevant way to the differentiation of the values of the utility funcions associated to the two typologies of firms. For this reason, the parameter η plays a less important role in the formation of the utility function value, and is not able to influence in a significant way the purchase choices of customers.
On the contrary, in the second model we have vendors very similar among them. They distinguish only for the way in which they sell their products with or without the availability of the source code. They are characterized by similar investment policies, which entail the

market positioning of products which differ not much from each other as regards price and quality.
Consequently, the values of the quality Q_i, the price P_i and the switching costs ρ do not contribute in a relevant way to the differentiation of the values of the utility functions associated to the two typologies of firms.
For this reason, the parameter η plays a very important role in the formation of the utility function value, and is able to influence in a significant way the purchase choices of customers.
Concerning all the other parameters, we can not underline any significant difference between the two study cases illustrated.

Part II

SOFTWARE DEVELOPMENT SIMULATION MODELS

Chapter 6

Introduction to the Software Development Simulation Models Proposed.

In this ***second part***, our attention is addressed to the simulation of some methodologies for software development. In particular, we conducted an analysis on Agile methodologies versus traditional methodology, and on Cloud Computing applied to the Global Software Development.

We applied the Dynamic Systems approach to realize a model for analyzing the dynamic behaviour of the adoption of Kanban and Scrum, versus a traditional software development process such as the Waterfall approach. Further, the same approach, through a version modified of the previous model, was applied for studying how Global Software Development can be facilitated using Cloud development environments, compared to a traditional development environment.

6.1 Background.

System dynamics modeling has been used in similar research, in which there are multiple and interacting software processes, time delays, and other nonlinear effects such as communication level, amount of overtime and workload, schedule pressure, budget pressure, rate of requirement change, and so on.

In the field of the Agile Methodologies many system dynamics models were introduced. The main goals of these researches aim to better understand the agile process and to evaluate its effectiveness. Most of the performed research was made on Extreme Programming (XP), or generic AMs. Other processes such as Scrum, however, are almost absent. For example, Chichacly in [7] investigated when AMs may work by using a System Dynamics Modeling and comparing AMs with a traditional waterfall process. In [72] the author explored whether agile project management has a unique structure or will fit within the generic conceptually formed system dynamic project management structures. An analysis of factors that impact on productivity during agile web development and maintenance phases was conducted by Xiaoying Kong et al. [33]. Another analysis published in [32] gives both theoretical insights into the dynamics of agile software development, and practical suggestions for managing these projects.

In [45] Brooks' Law and the effects of Pair Programming are presented by modeling the process through the system dynamics, and the advantages of this approach are shown.

Misic et al in [70] present a system dynamics model of two key practices of Extreme Programming: pair programming and pair switching, and task switching. A comparison between XP and a traditional approach is carried out and the conditions under which one approach appears to give better results than the other are outlined.

Also in [73] a model of XP software development process has been developed by using the system dynamics simulation techniques. The authors validated the process model with a typical XP project and studied the effects of XP by changing the adoption levels of XP practices.

In another paper, Wernick et al. [50] investigated the impact of pair programming on the long term evolution of software systems by using system dynamics to build simulation models which predict the trend in system growth, with and without pair programming.

In addition to these literature's sources, and regarding the model proposed to study how effective Cloud-based software development environments are for Global Software Development to facilitate the daily work, we can cite the works by Hossain et al.[15] and Hashmi et al.[54]. In [15] Hossain, Babar, Paik, and Verner discuss the use of Scrum practices in GSD projects, and identify key challenges due to global project distribution which restricts the use of Scrum. In [54] instead, the authors present the challenges encountered in globally dispersed software projects and propose to exploit Cloud Computing characteristics and privileges both as a product and as a process to improve GSD. In a more and more globalized world the relationship between culture and management of remote work is an anavoidable issue to face, so they proposed to exploit Cloud Computing both as a product and as a process to manage the many challenges in terms of culture, management, outsourcing, organization, coordination, collaboration, communication, development team, development process and tool. The authors identified GSD challenges and requirements that a cloud architecture could solve. For example, they affirm that cloud services, as a product, ensure interactions among different activities, while as a process it could allow resource sharing, infrastructure and application resources, but also software resources and business processes. Again the issue of geographically distance could be solved with Platform as a Service. It does not require any kind of downloads and installations: therefore used as a product it supports geographically distributed teams, instead when used as a process, it can help to overcome many limitations encountered in software evolution, reuse and deployment. The work which inspired our model is [3]: a practical experience in the application of some agile software development practices, as Scrum model, to Azure application development. Azure Services Platform is an application platform on the cloud and it offers PaaS capabilities, which allow application to be built and consumed from both on-premise and on-demand environments. This paper starts from several questions about the interactions between cloud computing and agile software development and attempts to discuss their potential advantages. In fact the authors show how setting up a development environment on the Azure platform helps enhance the agile practices.

Chapter 7

Simulating Kanban and Scrum vs Waterfall with System Dynamics

In this chapter we analyze the dynamic behaviour of the adoption of Kanban and Scrum, versus a traditional software development process such as the Waterfall approach. We use a system dynamics model, based on the relationships between system variables, to assess the relative benefits of the studied approaches. The model is simulated using a commercial tool available on the market: Vensim[1]. The proposed model visualizes the relationships among these software development processes, and can be used to study their relative advantages and disadvantages.

The model is built by using an analysis of feedback loops among the components of the processes, such as requirements, iterations, releases and so on, and through workflows and delays, to control their dynamics. In order to compare these processes, we start with a paradigmatic project with fixed requirements, expressed as a given number of features to be incrementally implemented.

Software processes simulation can be useful under various aspects: it can help to improve current processes or to enforce motivation for changes [6].

To our knowledge, this is the first time that Scrum and Kanban AMs are studied by using systems dynamics models.

7.1 An overview of Waterfall, Lean-Kanban and Scrum processes

In this section we take a look at the some software development approaches, to identify their relative strengths and weaknesses.

The Waterfall model was introduced by Royce in 1970. It is a traditional "heavyweight" software development methodology in which all process phases (planning, design, development, testing and deployment) are performed in a sequential series of steps. Each phase starts only when the previous one has ended. It is possible to step back to the previous phase,

[1]http:
www.vensim.com

but it is not possible to go back in the process, for instance in order to accomodate a substantial change of requirements. This methodology requires to define a stable set of requirements only during the phase of requirements definition, and feedbacks to previous stages are not easily introduced.

In response to this kind of rigid, hard to follow methodology, Agile Methodologies, so named in 2001 in the Agile Manifesto [68], have been introduced. Among them, Scrum and Lean-Kanban are Agile process tools [20] based on incremental development. They both use pull scheduling and emphasize on delivering releasable software often.

The original term Scrum comes from a study by Takeuchi and Nonaka [65] that was published at 1986 in the Harvard Business Review. In 1993 Jeff Sutherland developed the Scrum process at Easel Corporation, by using their study and their analogy as the name of the process as a whole. Finally, Ken Schwaber [58], [57] formalized the process for the worldwide software industry in the first published paper on Scrum at OOPSLA 1995. Scrum [59] is a simple agile framework, adaptabile also to contexts different from software development [42]. Scrum has three roles (Product Owner, Scrum Master, Team), three ceremonies (Sprint Planning, Sprint Review, and Daily Scrum Meeting), and three artifacts (Product Backlog, Sprint Backlog, and Burndown Chart). Adopting Scrum implies to use timeboxed iterations and to break the work into a list of smaller deliverables, ordered according to a priority given by the Product Owner. Changes to requirements are not accepted during the iteration, but are welcomed otherwise. Scrum projects are organized with the help of daily Scrums: 15 minutes update meetings, and monthly Sprints, or iterations, that are designed to keep the project flowing quickly. Generally, at the end of every iteration the team releases working code, and a retrospective meeting is held also to look for ways to improve the process for the next iteration.

Lean software development is a translation of Lean manufacturing [29] to the software development domain. The Lean approach emphasizes improving the flow of value given to the customer, eliminating waste (Muda), and consider the whole project, avoiding local optimizations.

Kanban is a Japanese term that translated literally means visual (Kan) and card or board (ban). Adopting Kanban means to break the work into work items, to write their description on cards, and to put the cards on a Kanban board, so that the flow of work is made visible to all members of the team, and the Work in Process (WIP) limits are made explicit on the board. The Kanban board provides a high visibility to the software process, because it shows the assignment of work to developers, communicates priorities and highlights bottlenecks. One of the key goals of Lean-Kanban approach is to minimize WIP, so that only what is needed is developed, there is a constant flow of released work items to the customer, and developers focus only to deliver a few items at a time. So, the process is optimized and lead time can be reduced.

In a nutshell, Scrum and Lean-Kanban approaches are both agile processes aiming to quickly adapt the process by using feedbacks loops. In Lean-Kanban the feedback loops are shorter, and work does not flow through time-boxed iterations, but flows continuously and smoothly. Kanban is less prescriptive than Scrum and it is able to release anytime, while Scrum will release new features only at the end of the iterations. Moreover, in Scrum it is not possible to change the requirements in the middle of the sprint.

7.2 Model structure

Our model uses a simplified version both of the Waterfall process and of the Scrum and Lean-Kanban approach, so that its structure is easier to understand and to model. The Waterfall process has been selected because it represents the most opposite methodology to agile processes, and Scrum and Lean-Kanban have been selected because there are not enough scientific empirical studies about them. The objectives of the study are to identify relationships and mechanisms within a software project, so the model is focused on the tendencies of the simulation results and not on specific, quantitative aspects.

According to SD modeling, our model is represented in terms of stocks, and flows in an out of them. The first step has been to identify what are the flows in the main process, which in our case are the project requirements. The auxiliary variables control the "valves" acting on the alternative outflows of each stage.

Based on what normally happens, we made the assumption that a project is developed by a small team – let us suppose 10 developers. The software development process is conceptualized as transforming a initial stock of requirements that need to be developed (Original Work to Do) to a stock of developed requirements (Live). The initial stock of requirements, and the team size, is the same for all simulated processes. A requirement is defined as a set of functionalities to implement. The size of the project for the three processes is estimated as 210 requirements (features in Scrum and Lean-Kanban) to develop that have unit weight and equal size.

We have implemented the three processes with a different subdivision of the work:

- **Scrum**: the work has been split into a set of Sprint backlogs, each including a random number (in the range 15-21) of features selected from a Gaussian distribution. These backlogs are developed during short fixed-length iterations of 2 weeks;
- **Lean-Kanban**: the work has been split in a set of iterations. For each iteration, a random number of 6 - 10 features have been selected from a Gaussian distribution;
- **Waterfall**: the work is not split, but all the features to develop are placed in "Selected Requirements", with a stock of requirements to do that we assume to be 210 for the entire project.

In order to simplify the model, all phases of planning, design, coding, testing and similar have been merged into just one development phase, represented by the *requirements development rate* valve. In Scrum and Waterfall processes, the planning phase has been taken into account by introducing some delays equal to the time spent to planning.

The *requirements development rate* is the speed at which requirements flow to "Fraction Work Done". This rate is determined by the team's productivity, by the number of developers and by the error rate and it is defined by the following equation:

requirements development rate = IF THEN ELSE (switch=1, IF THEN ELSE(Selected Requirements >0.05:AND:Selected Requirements $\geq$ productivity $\times n^o$ developers,(productivity $\times n^o$ developers) $\times$ (1-error in Kanban), IF THEN ELSE (Selected Requirements >0.05:AND:Selected Requirements < productivity $\times n^o$ developers,Selected Requirements $\times$ (1-error in Kanban), Selected Requirements)),IF THEN ELSE(Selected Requirements > 0.05:AND: Selected Requirements $\geq$ productivity $\times n^o$ developers, (productivity $\times n^o$ developers) $\times$ (1-error in sprint

Scrum or in Waterfall), IF THEN ELSE (Selected Requirements > 0.05:AND: Selected Requirements < productivity × n^o developers, Selected Requirements × (1-error in sprint Scrum or in Waterfall), Selected Requirements)))

We set an average productivity of 0.025 requirements per hour (the number of requirements developed in one hour) because we assume that a developer is able to implement a requirement in about 5 days. Moreover, the fewer errors there are in the process, the sooner the project will be finished.

As mentioned before, the requirements for each approach are developed in the "requirements development rate" valve. In our model, for each iteration, only a fraction of the requirements referred to by the "Selected Requirements" variable is completed because a fraction of the work is done incorrectly due to three types of error: *effect of uncertain customer requirements, problem in the software design, bug introduced during the development.*

As requirements are implemented, they flow into the "User Acceptance Testing" variable, representing the feedback given by the customer. This phase introduces another delay in the process, with a different weight according to the simulated software process. In general terms user acceptance testing is considered to be an essential step before the system is eventually accepted by the end user. If the acceptance test is successfully passed, then the requirements are accepted and are considered completed (the accepted requirements flow into the "Live" level). Otherwise, a rework must be performed, which includes a delay due to the correction.

A simplified version of the model structure with stocks and flows is shown in Fig. 7.1.

In Table 7.1 we report the initial parameters we used. Note that the model data have been gathered through a series of interviews with software development professionals in various organizations, and through a review of the literature.

Consequences of errors and delays on software development

In our model, the length of the iteration is affected by three main effects: delays, errors and rework on software already developed. These effects influence project outcomes and determine the system development speed. In our model, an important role is assigned to the delays. Fig. 7.3 shows a simplified view of the model.

Our simulator has been designed with the goal to highlight the main differences among the Waterfall, Scrum and Lean-Kanban approaches. Waterfall is very prescriptive and assumes specific sequential phases: each phase must be clearly ended before the next may start. All phases must be completed before you can start the process again. Moreover, if requirements change during project development, the waterfall model requires the completion of a full cycle before they can be revisited. The planning phase in our Waterfall model implies a delay equal to *waterfall delay*. The other phases have been modeled in *requirements development rate.*

Scrum and Lean-Kanban are agile and lean processes, which are less prescriptive than Waterfall. Moreover, Scrum is more prescriptive than Lean-Kanban. In particular, Scrum prescribes roles such as the role of the Product Owner – a single person with a final authority representing the customer's interest in backlog prioritization and requirements questions.

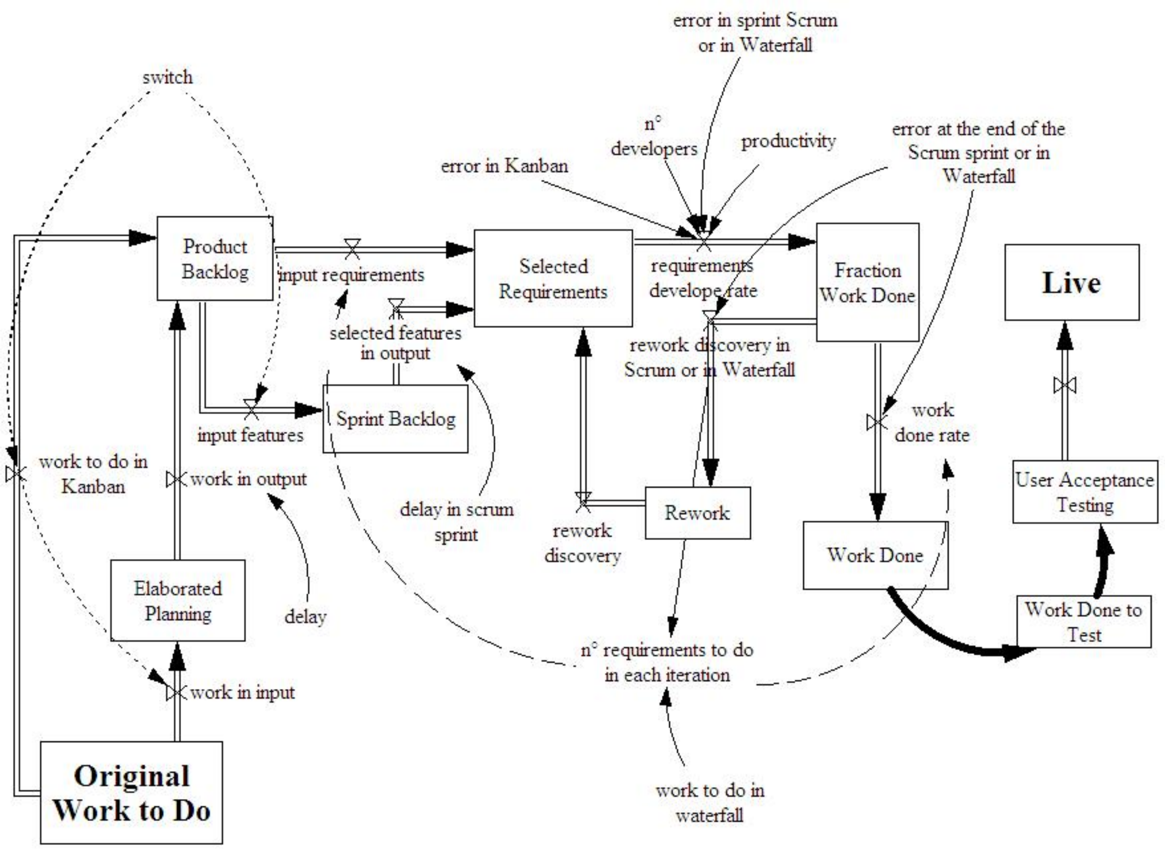


Figure 7.1: Simplified version of the structure of the proposed model.

Moreover, a Sprint Planning Meeting and a Sprint Retrospective Meeting are prescribed, respectively to plan the iteration and at the end of every sprint. The sprint review meeting aims to discuss what went well and what to improve in the next sprint. In our model, these meetings and the activity of the Product Owner are modeled through the following variables: *delay for Sprint Planning Meeting, delay for Retrospective Meeting* and *Scrum delay*.

As regards the effects of errors and of rework, Fig. 7.2 shows the causal loop effect on the error rate. We identify a reinforcing feedback loop (positive feedback) where cause and effect can be circular and an effect strengthens its own cause, or where sometimes a cause has multiple opposing effects. For example, in the our case, the *effect of uncertain customer requirements* reinforces *error rate* and at the same time *benefits of keeping things simple* decreases *error rate.*

As mentioned before, a fraction of the work is considered to be done incorrectly due to three types of error: *effect of uncertain customer requirements, problem in the software design, bug introduced during the development.* In Lean-Kanban process, these three errors are discovered and corrected in the *requirements development rate* valve, instead in Scrum and in Waterfall only the *bug introduced during the development* error passes through in the *rework discovery in Scrum or Waterfall* valve characterized by the following equation:

rework discovery in Scrum or in Waterfall = IF THEN ELSE(switch= 0:OR:switch=2,IF THEN ELSE(Selected Requirements < 0.05, Fraction work done ×(error at the end of the

Table 7.1: *Initial Parameters of the proposed model.*

	Waterfall	**Scrum**	**Lean-Kanban**
n^o features in Scrum	-	Integer[Random Normal (15 , 21)]	-
n^o selected features in Kanban	-	-	Integer[Random Normal (6 , 10)]
Scrum delay	-	2 hours	-
Waterfall delay	120 hours	-	-
Delay for meeting of sprint planning	-	Integer[Random Normal (4 , 8)]	-
Delay for retrospective meeting	1 hours	-	-
Bug introduced during the development	Random Normal (0.0071, 0.0095)	Random Normal (0.0071, 0.0095)	Random Normal (0.0071, 0.0095)
Problem in the software design	0	Random Normal [0.001, 0.0014]	Random Normal (0.001, 0.0014)
Effect of uncertain customer requirements	0	Random Normal (0.0047, 0.0071)	Random Normal (0.0047, 0.0071)

Scrum sprint or in Waterfall), 0),0)

The other two errors can be discovered only at the end of the iteration.

In the Waterfall model, the error rate at the end of the iteration increases because the releases are not frequent and the work size in an iteration is more heavyweight than in Scrum and Kanban, so according to [12] *"an error introduced during the requirements phase, but not discovered until maintenance, can be as much as 100 times more than that of fixing the error during the early development phases"*, the variable (*effect of uncertain customer requirements* and the variable *problem in the software design*) are proportional to the *proportional factor* variable.

In Fig. 7.4 the time trends of the rework in Scrum and in the Waterfall approach at the end of the iteration are reported, respectively.

7.3 Results

We mentioned that Lean-Kanban approach does not prescribe roles and meetings, and the number of requirements to implement at any given time is very small. So also our proposed Lean-Kanban model limits the work in progress and minimizes the lead time. In Fig. 7.6 and in Fig. 7.7 we can observe that very many requirements are implemented in a Waterfall iteration. The number of selected requirements in Lean-Kanban and in Scrum is much smaller than in Waterfall model, while in Kanban this number is smaller than in Scrum. In fact, "Selected Requirements" variable is equal to "Original Work" after a time interval equal to the *Waterfall delay*. This variable highlights the speed with which the requirements flow from "Selected Requirements" level to "Fraction Work Done" level. For each approach we report the time trend of "Selected Requirements" that is characterized by a waveform that

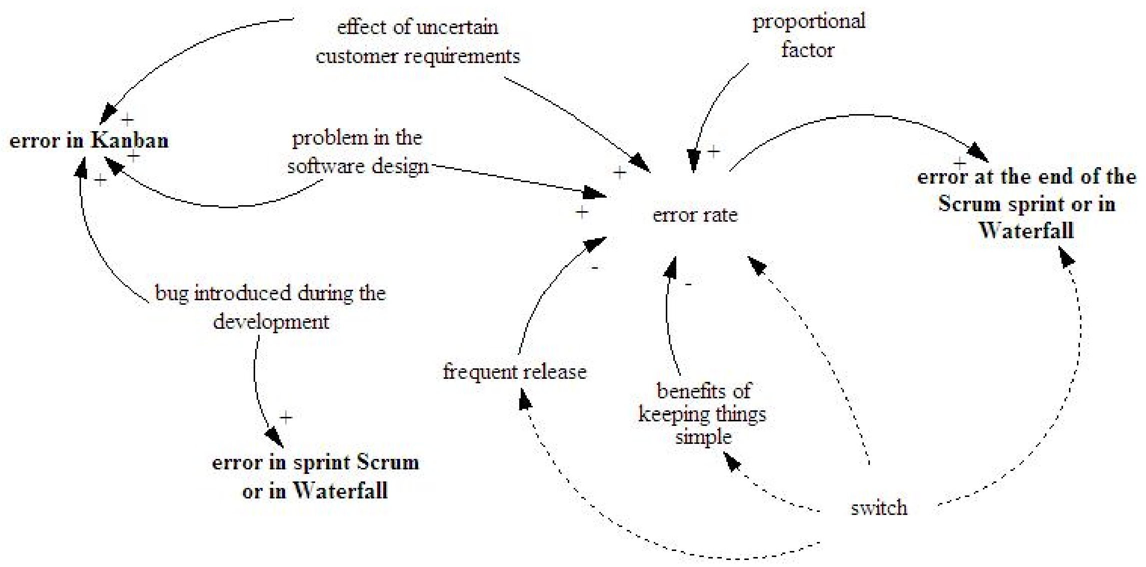

effect of uncertain customer requirements
proportional factor
error in Kanban
problem in the software design
error rate
error at the end of the Scrum sprint or in Waterfall
bug introduced during the development
frequent release
benefits of keeping things simple
error in sprint Scrum or in Waterfall
switch

Figure 7.2: Causal loop diagram of error rate.

is repeated periodically during the simulation. The waveform's period coincides with the length of each iteration, and we can observe that the initial amplitude of the waveform in Waterfall approach is larger than in Scrum and Lean-Kanban approach.

Moreover as can be seen in Figure 7.4 (a) in the Scrum approach the amount of rework at the end of the iteration is lower than the amount of rework in Waterfall approach (see Fig.7.4 (b)). In agreement with this is the fact that in Scrum approach the rework is calculated with a smaller number of requirements with respect to the amount of requirements in Waterfall approach. In fact if in the proposed model the rework is calculated with "n^o Requirements to Do in each Iteration" equivalent to the "Number of Features in Scrum" in Scrum approach (as mentioned in Scrum the number of requirements can vary between 15 and 21), instead in the Waterfall process the "Original Work to Do" is not broken into iterations, and so the rework at the end of the iteration is calculated with a number of requirements equivalent to "n^o requirements to do in each iteration" which coincides with "work to do in Waterfall" equal to "Original Work to Do" and then to 210 requirements.

We think that the outputs obtained by our model are sensible if compared to real world data. Fig. 7.8 and Fig. 7.9 show a simulation fitting the behaviour of a real project. In Fig. 7.8 we show the time trend of the work done during the tree approaches before the software goes "Live" and the acceptance testing has been carried out, while in Fig. 7.9 after the customer requirements has been validated.

The proposed model and the analysis of the data suggest that Lean-Kanban approach is more efficient than the other two approaches, thanks to a software development mechanism that allows frequent releases and the division of the work in very small time chunks.

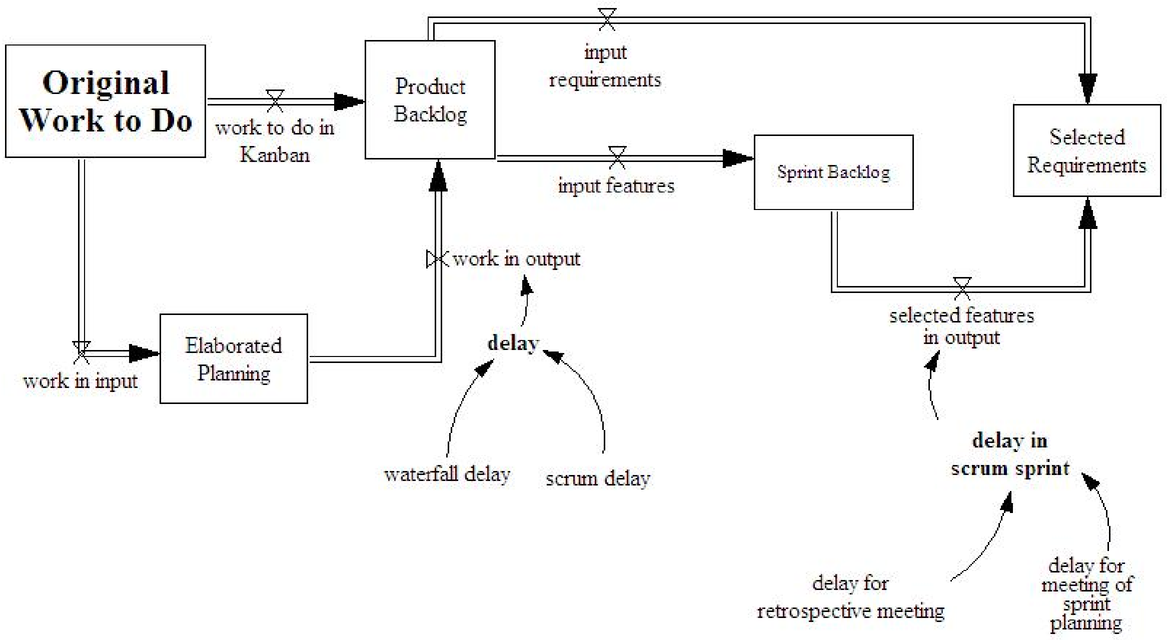


Figure 7.3: Delays in the proposed System Dynamic model.

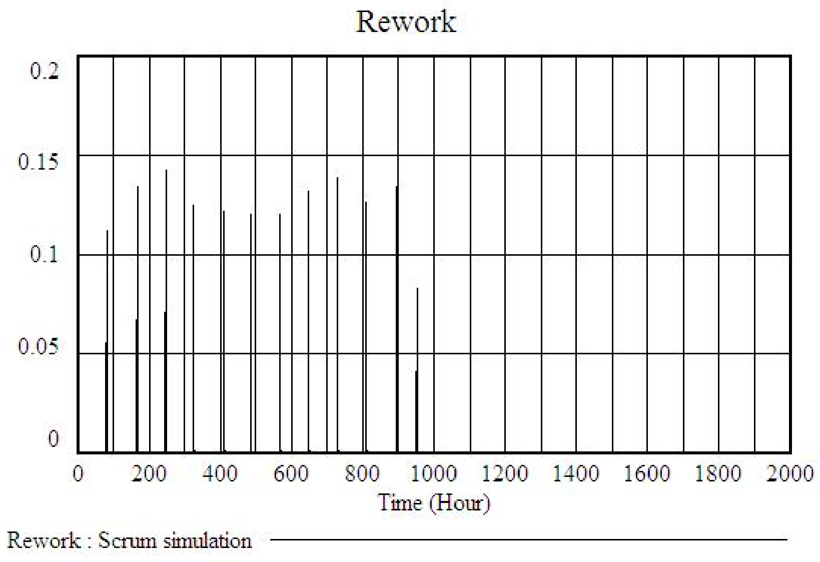


Figure 7.4: *Rework in Scrum approach*

7.4 Summary.

We developed a simplified system dynamics model for describing the behaviour of three different approaches to the software development under similar starting conditions. We compared by means of simulation techniques a heavy and prescriptive approach, Waterfall, with two agile and less prescriptive process tools, Scrum and Lean-Kanban.

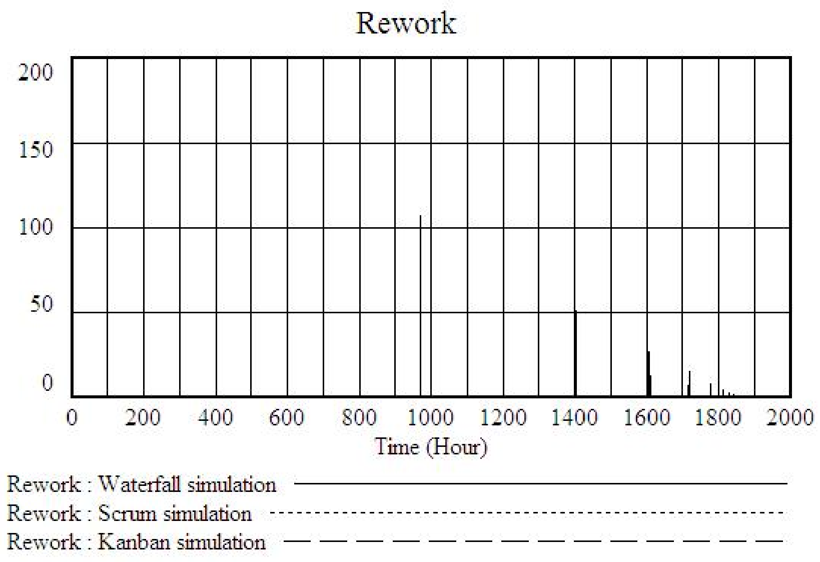


Figure 7.5: *Rework in Waterfall*

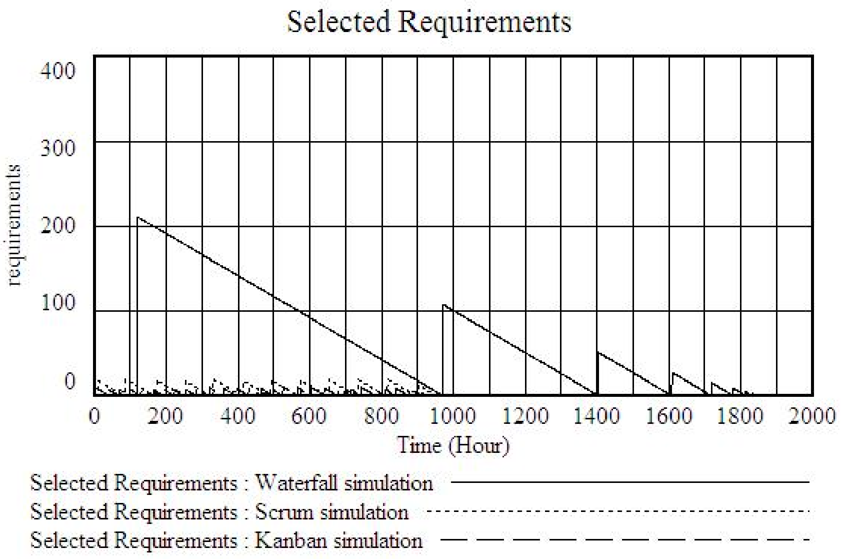


Figure 7.6: Selected Requirements in the three approaches.

In order to avoid a too complex model, all the variables influencing the processes were not included in our simulation, so our study has been carried out under some limiting assumptions that could threaten its validity. But this is a first model and it can be considered as a valid starting point for further studies. We described some strengths and weaknesses of three software process methods by modeling their environment with a continuous-time simulation tool. Although Lean-Kanban is well known in software development processes, it has not yet investigated in depth in research works. In our model, the Kanban workflow was managed through an effective control mechanism to limit the work in progress and minimize the lead time. One of the advantages of this approach is that the work is better con-

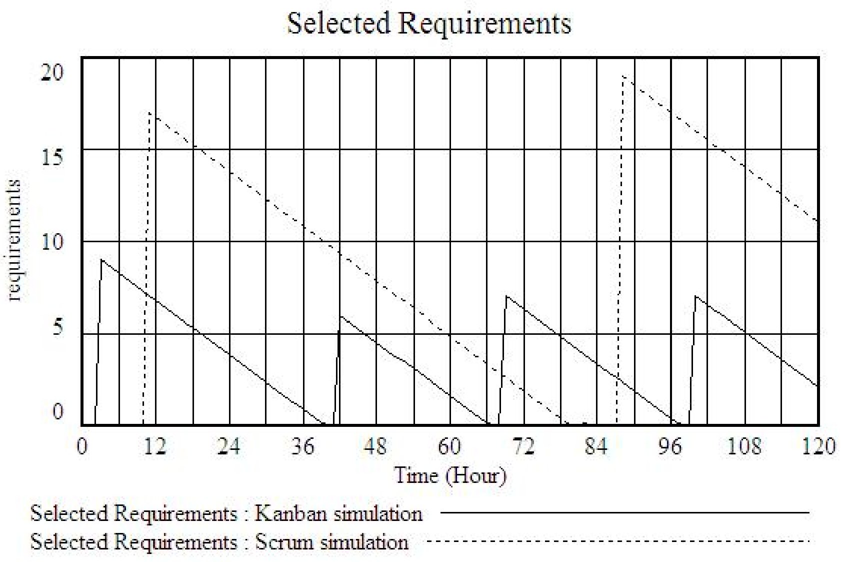


Figure 7.7: Selected Requirements in Scrum and Lean-Kanban.

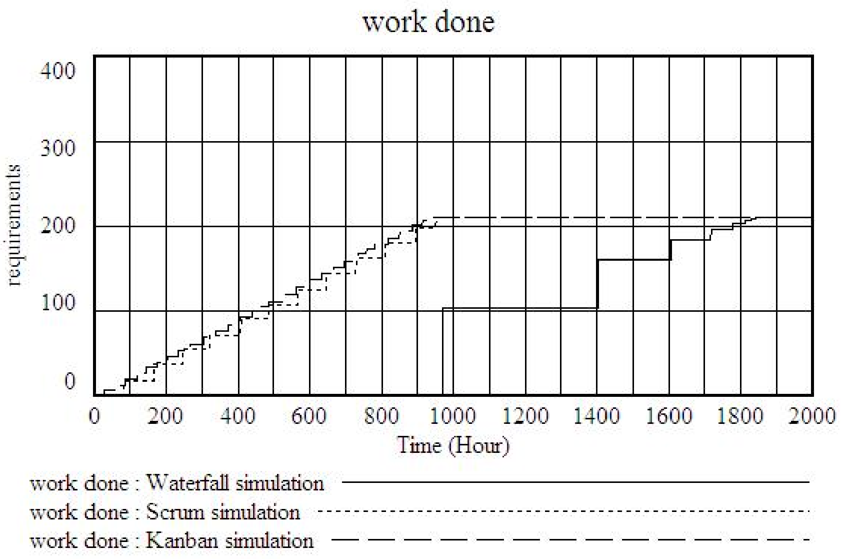


Figure 7.8: Work Done.

trolled, so that the effects of errors are kept limited. On the contrary, in the Waterfall case often projects may fail to complete due to the difficulty to correct errors, including errors in requirements.

The resulting behaviour of the simulation model presented is quite realistic, and its behaviour is the same also by changing the project size. However, this is a first model that needs to be further elaborated and validated, for example by adding new variables or new relationships among factors. This will be the subject of our future work, which will also include studies to empirically validate the simulation results using data from real projects.

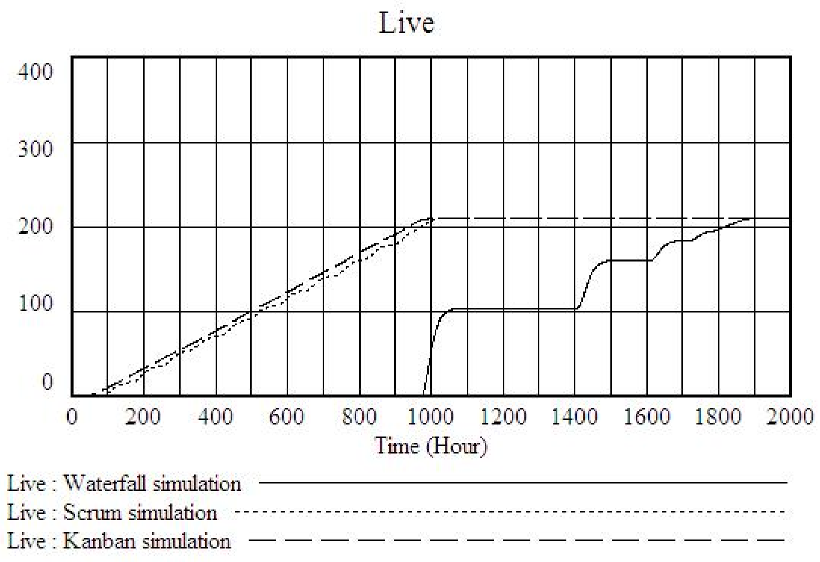


Figure 7.9: Live.

Chapter 8

A Model for Global Software Development with Cloud Platforms.

Cloud Computing (CC) is a technological phenomenon that is becoming more and more important. Also Small and Medium Enterprises (SMEs) can increase their competitiveness by taking advantage of CC. This new computing approach promises to provide many advantages, and many SMEs are encouraged to use it. However, CC is still in its early stage – for this reason we think that it is very important to study and assess its impact on SMEs' management processes.

In this chapter we propose a model for studying how Global Software Development can be facilitated using Cloud development environments, compared to a traditional development environment. We use system dynamics to model and simulate how effective Cloud-based software development environments are for Global Software Development performed by a SMEs. Both studied environments assume a development process based on Scrum agile methodology. The proposed model could be used as a tool by the project managers to understand how Cloud Development Environments might facilitate Global Software Development.

Traditional software applications are based on a model with large upfront licensing costs and annual evergreen support costs. On the contrary, on-demand applications are based on a recurring subscription fee and the cost may increase as the usage of the application increases; in this approach, the possession of the physical location, of the hardware and of the system maintenance is in charge of cloud providers. Specifically, there is a number of issues that need to be considered. We compare two development environment, one on-premise, and the other on-demand, both set up for agile development models, in order to perform Global Software Development (GSD). Among available Agile Methodologies, we choose Scrum due to its popularity.

Scrum was proposed as a possible solution for quickly responding to changing customer requirements, without compromising the quality of the code, and is presently the most used Agile Methodology (AM) [69].

The model proposed is built by using an analysis of feedback loops among the components of the processes, such as requirements, iterations, releases and so on, and through workflows and delays, to control their dynamics. We simulated the models using a commercial tool available on the market: Vensim[1]. Software processes simulation can be useful in various respects – it can improve current processes, or enforce motivation for changes [6].

In order to compare these models, we assume a software project with fixed requirements to be incrementally implemented. To our knowledge, this is the first time that GSD is studied by using system dynamics models.

8.1 An overview of Global Software Development.

A common definition of Global Software Development is a software development process at geographically separated locations. For this reason, GSD involves communication for information exchange, coordination of teams, activities and artifacts so they contribute to the overall objective, and finally control of teams. Many challenges meet into GSD [15] , [54] – these are geographic, cultural, linguistic and temporal. The distance and the lack of overlapping working hours create a negative impact on software projects, create problems in the knowledge transfer, and as a consequence communications gaps or ambiguity on technical aspects may occur. Cultural diversities may bring to an unequal distribution of work, lack of trust and fear, from which cost increases, poor skill management and reporting issues may arise. Linguistics and temporal diversities can instead lead to issues in knowledge transfer, communication and project visibility.

The GSD can be facilitated using the Cloud. CC is a delivery model for software, platforms and infrastructures. Cloud providers have got the possession of physical location, hardware, and system maintenance. Enterprise users access cloud services via Internet from anywhere and at any time. Users usually pay a subscription fee, and can run on a robust infrastructure, a single instance of system. Cloud services are in fact delivered from a “multi-tenant" system; there is a single instance of software running, but many individual or enterprise customers use this system concurrently for their own necessities.

Cloud Computing contains 3 different paradigms ([51]):

- Software as a Service (SaaS),
- Platform as a Service (PaaS),
- Infrastructure as a Service (IaaS).

. The first is a cloud hosting model where the providers offer a complete end-user application. The second is a model that provides a full or partial application development environment. Providers offer in addition to the hardware, a platform including OS, middleware and runtime environment. Finally the last, Infrastructure as a Service is a model that provides a full computer infrastructure, server, router, storage, hardware and virtualization software. Moreover these cloud deployment models can be Private, Public or Hybrid depending on the scope of the cloud solutions. In the public model the cloud computing services are accessible to a large number of end-users, on the contrary in the private model the companies realize a cloud computing environment that allows to store the data within their operational structure with obvious advantages in terms of security and privacy. In this last case the cloud services are accessible only by authorized end-users. The third model is a system that is in the middle between the two models described above. It allows to enjoy the benefits of public

[1]www.vensim.com

model especially in terms of costs and scalability, and at the same time to have the guaranteed standards of management and security typical of the private models.

GSD seems be perfectly integrable with CC. In fact, GSD can be potentially improved thanks to the main characteristics of Cloud Computing: virtualization, reduced cost, scalability, infrastructure, performance and Multi Tenancy Support [54]:

- *Virtualization*: it allows to enhance the infrastructure to satisfy growing demand for services;
- *reduced cost*: Cloud services are based on the "pay as you go" model, so the subscribers pay only for the resources they use;
- *scalability*: it allows dealing successfully peak demand and avoid under-utilization of computer resources;
- *infrastructure*: the underlying infrastructure is invisible to the customers, who do not have to worry about it;
- *performance*: each consumer can set up on the Cloud the level of performances more suited to her needs;
- *multi tenancy support*: Cloud services are delivered from a multi-tenant system; a single istance of the system is running, but many consumers can use the system, as their workloads are isolated to ensure security and privacy of their data.

The positive implications that Cloud Computing can provide, with respect to traditional software development environments are many. Among the benefits linked to Cloud services, ([3], [55], [11], and [75]), we can name:

- no need to invest upfront in infrastructure;
- no hardware or software required for cloud services;
- no maintenance of the infrastructure that runs development applications;
- optimum utilization of in-house IT resources;
- building only what is needed, eliminating anything that does not add value;
- faster time to market;
- pay per use;
- easy setup and deployment of applications, uploading them on the Cloud;
- easy integration with other enterprise solutions;
- efficient collaboration among teams during all software development phases;
- access to a global environment;
- highly customized environment;

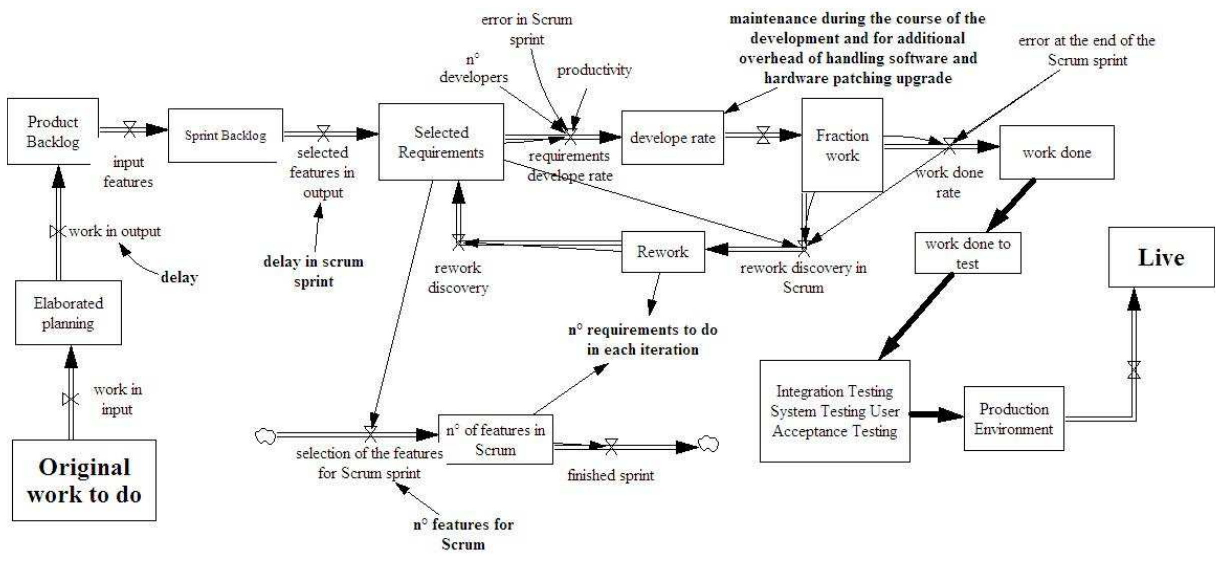

error in Scrum sprint
n° developers
productivity
maintenance during the course of the development and for additional overhead of handling software and hardware patching upgrade
error at the end of the Scrum sprint
Product Backlog
input features
Sprint Backlog
selected features in output
Selected Requirements
requirements develope rate
develope rate
Fraction work
work done rate
work done
work in output
delay
delay in scrum sprint
rework discovery
Rework
rework discovery in Scrum
work done to test
Live
Elaborated planning
n° requirements to do in each iteration
work in input
Integration Testing System Testing User Acceptance Testing
Production Environment
Original work to do
selection of the features for Scrum sprint
n° of features in Scrum
finished sprint
n° features for Scrum

Figure 8.1: Simplified version of the proposed model.

- costs savings.

The model proposed in Section 8.2 helps to highlight these issues, offering a way to estimate the potential benefits. The simulator developed in Vensim environment is a simple, customizable tool that each enterprise can set up in order to study how these benefits can influence its development process.

8.2 Model structure

Our model uses a simplified version of the Scrum approach in both on-premise and on-demand development model, so that its structure is easier to understand and to model, according to SD modeling, and as reported in [39] and in the previous chapter, our model is represented in terms of stocks and flows in an out of them.

A simplified version of the model structure with stocks and flows is shown in Fig. 7.1.

In our model, the time to finish the work *Original Work to Do* is affected by two main effects: delays and errors.

These effects influence project outcomes and determine the system development speed. In our model, an important role is assigned to delays. In traditional development environments, at the beginning of a project development lifecycle, considerable time and effort are required for procuring hardware and software licenses, and during the development additional time might be spent for upgrading this environment. On the contrary, in Cloud development environments, the infrastructure is readily available, and system maintenance and system updates of the cloud server will be taken care of by the cloud providers, and not by the developers. Moreover, Production and Testing Cloud environments are accessible anytime, anywhere: any team member and user can work and build applications referring just to one location, with no need to coordinate multiple locations. In this way, significant time and effort will be saved, and users will use all their resources for creating value for their business.

In our model, time and effort spent at the beginning of project development lifecycle are modeled by the following variables: *setting up infrastructure hardware and software li-*

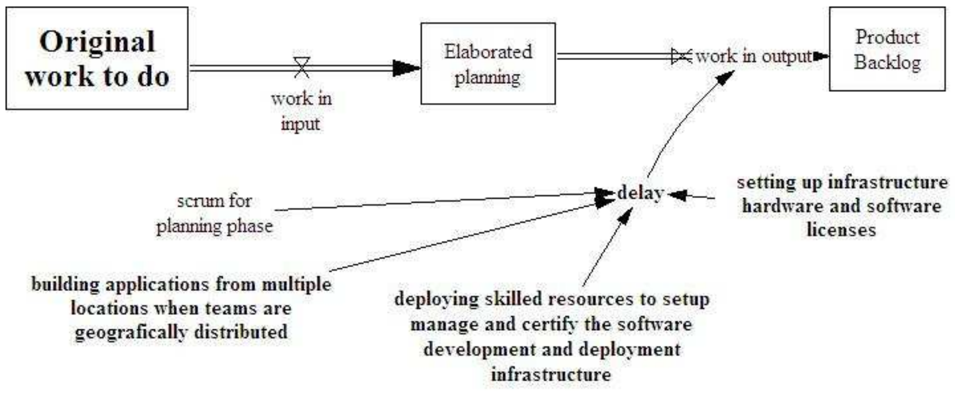

Original work to do
work in input
Elaborated planning
work in output
Product Backlog
scrum for planning phase
delay
setting up infrastructure hardware and software licenses
building applications from multiple locations when teams are geografically distributed
deploying skilled resources to setup manage and certify the software development and deployment infrastructure

Figure 8.2: Delays in the planning.

censes, deploying skilled resources to setup, manage and certify the software development and deployment infrastructure, building applications from multiple locations when teams geographically distributed are added. These variables, some of which were taken from the work of Dumbre et al. [3] lead to delays. Another variable should be added to these ones, for planning and creating the Backlog. This variable is indicated as *scrum for planning phase*. These four variables are reported in Fig. 8.2.

Time and effort for the maintenance of traditional environments influence the project during the development phase. This may happen during the Scrum sprint, and is modeled with the variable called *maintenance during the course of the development and for additional overhead of handling software and hardware patching upgrades*, shown in Fig. 8.3.

Regarding what affects the length of Scrum iteration, Scrum prescribes roles such as the Product Owner – a single person representing the customer's interest who is in charge of backlog prioritization, and who shall answer to questions about requirements. Moreover, a Sprint Planning Meeting, a Sprint Retrospective Meeting and Scrum of Scrums meeting are prescribed, respectively to plan the iteration, at the end of every sprint, and to coordinate more teams that work at geographically separated locations. The sprint review meeting aims to discuss what went well and what to improve in the next sprint. In our model, these meetings and the activity of the Product Owner are modeled through the following variables: *Sprint Planning Meeting, Retrospective Meeting, Scrum delay* and *Scrum of Scrums* illustrated in Fig. 8.4.

As regards the effects of errors, Fig. 8.5 shows the causal loop effect on the error rate. We identify a reinforcing feedback loop (positive feedback) where cause and effect can be circular and an effect strengthens its own cause, or where sometimes a cause has multiple opposing effects. For example the *effect of uncertain customer requirements* reinforces *error rate.*

As mentioned before, a fraction of the work is considered to be done incorrectly due to three types of error: *effect of uncertain customer requirements, problem in the software design, bug introduced during the development.* Only the *bug introduced during the development* error passes through the *rework discovery in Scrum* valve characterized by the following

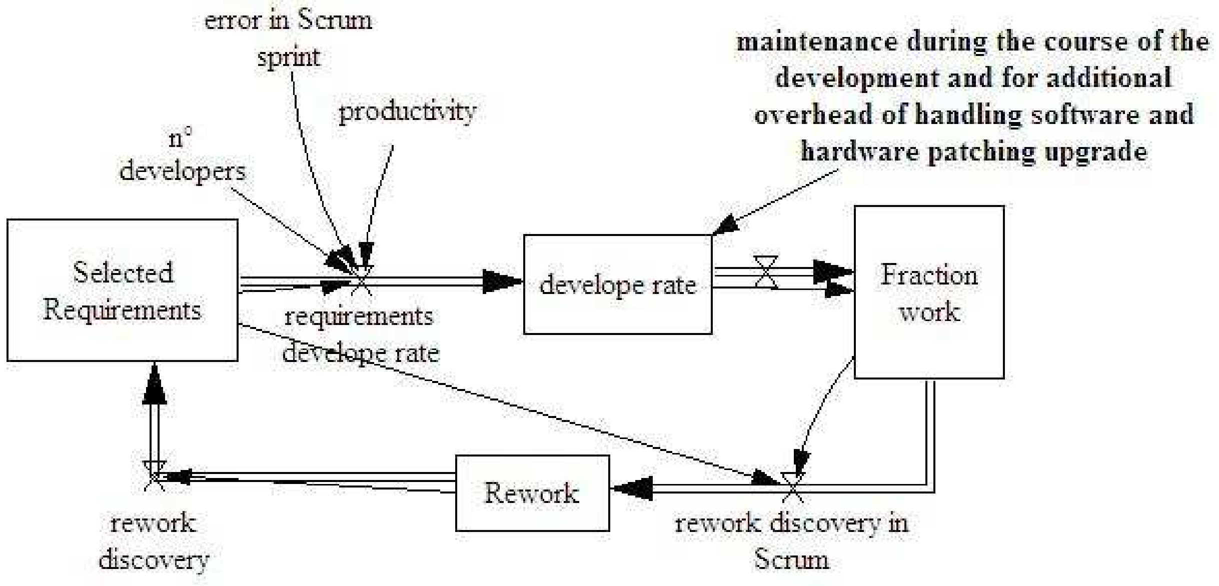

error in Scrum sprint
productivity
n° developers
maintenance during the course of the development and for additional overhead of handling software and hardware patching upgrade
Selected Requirements
requirements develope rate
develope rate
Fraction work
Rework
rework discovery
rework discovery in Scrum

Figure 8.3: Development phase.

equation:

rework discovery in Scrum = IF THEN ELSE(Selected Requirements < 0.05, Fraction work done ×(error at the end of the Scrum sprint), 0)

The two other errors can be discovered only at the end of the iteration.

Finally, we analyze the time and effort spent in the different test environments, and in the production environment. Under traditional development environment, much time and effort are spent to write verbose installation scripts or release notes, and for the setup of system testing, integration testing and user acceptance testing, with the aim to obtain a product released under rigorous test and validation. Moreover, the developers must often deliver rapid prototyping and demos to obtain immediate feedback from the end user. This may become a real challenge if the user and the developers work in different locations.

In cloud development environments, instead, these steps are simpler than in a traditional environment. The deployment process is simplified, there is no need of any separate packaging efforts; to pass from development environment to testing environment, and from here to production environment does not require any additional step. Prototypes and demos can be made accessible immediately to customers for eliciting feedback in a short time. Code can pass from one environment to another without writing deployment script to set up the application in the respective environments. What has just been said is modeled by two variables: *creating and managing different test environments* and *creating and managing production environment prototyping and demos* introduced when the requirements are moved from *work done to test* to a different testing environment, and then are deployed to

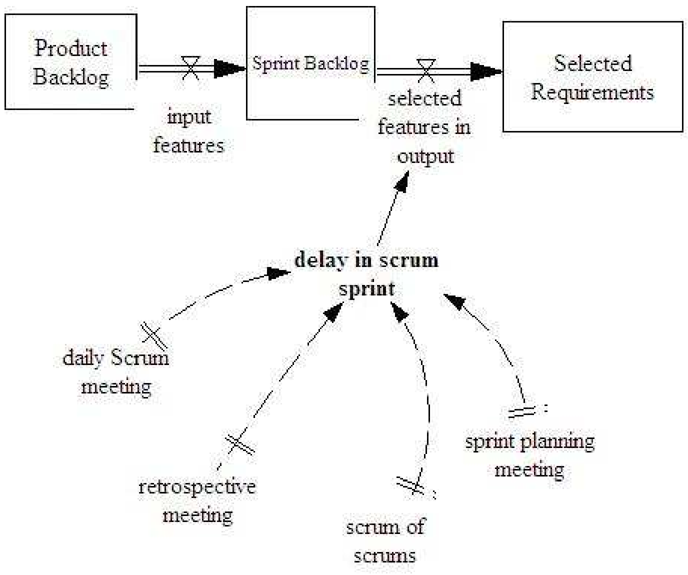


Figure 8.4: Delay for meetings.

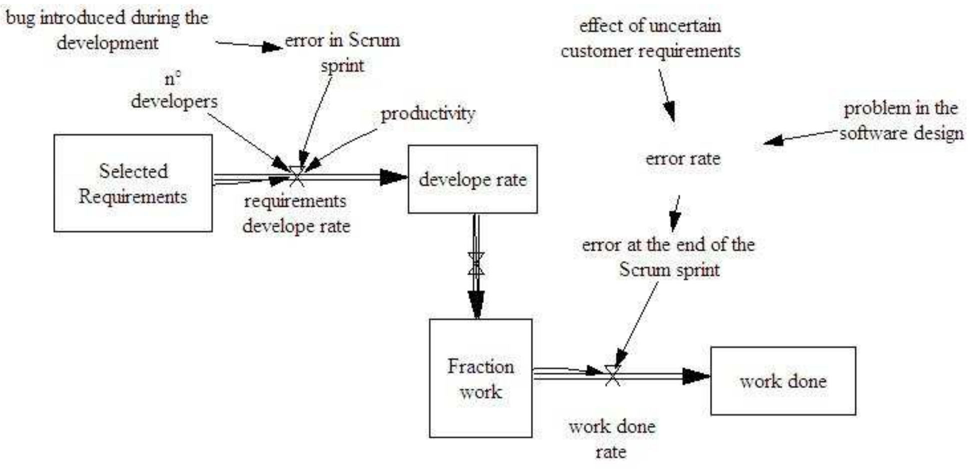

bug introduced during the development
error in Scrum sprint
n° developers
productivity
effect of uncertain customer requirements
problem in the software design
error rate
Selected Requirements
requirements develope rate
develope rate
error at the end of the Scrum sprint
Fraction work
work done rate
work done

Figure 8.5: Causal loop diagram of error rate.

production environment. The latter variables are shown in Fig. 8.6.

All the variables described above represent time and effort spent in the development process, leading to process delays, which mean further overheads of cost and effort.

8.3 Results

The system dynamic simulation model described in the previous sections has been proposed for analyzing how the Cloud approach can facilitate the GSD with respect to the traditional approach. We studied the two different approaches, and for switching between them, we introduced in the model a variable called *switch*. If *switch* value is equal to 0, the mod-

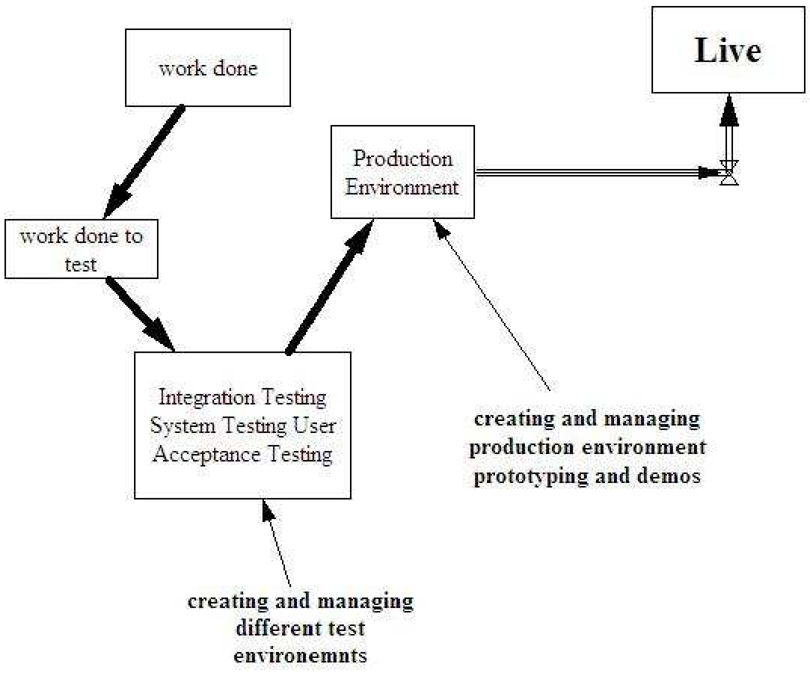

work done
work done to test
Integration Testing System Testing User Acceptance Testing
Production Environment
Live
creating and managing production environment prototyping and demos
creating and managing different test environemnts

Figure 8.6: Delay for testing and production environments.

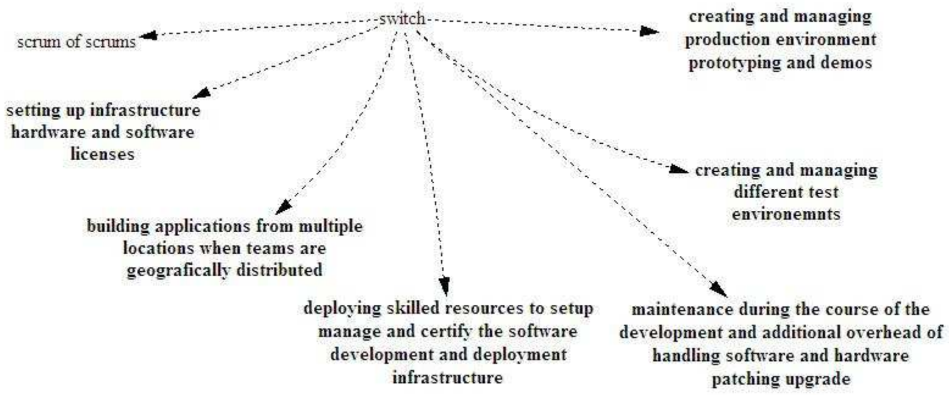

switch
scrum of scrums
creating and managing production environment prototyping and demos
setting up infrastructure hardware and software licenses
creating and managing different test environemnts
building applications from multiple locations when teams are geografically distributed
deploying skilled resources to setup manage and certify the software development and deployment infrastructure
maintenance during the course of the development and additional overhead of handling software and hardware patching upgrade

Figure 8.7: Work done.

eled environment is the Cloud, if *switch* is equal to 1 the environment is traditional. In figure 8.7 we report the variable *switch* and all the variables that assume different values in the two type of systems studied.

The results have been obtained setting up the initial variables according to the considerations described in previous Sections and it is just an example showing how our simulator works, and what can be found by using it.

In figure 8.8 we report the time trend of the work done, and we can notice how the development rate of software is higher in Cloud system than Traditional system.

Our model was simulated implementing only the variables mentioned in Section 8.2, because we chose an hypothetical company with special needs, but it is important to note

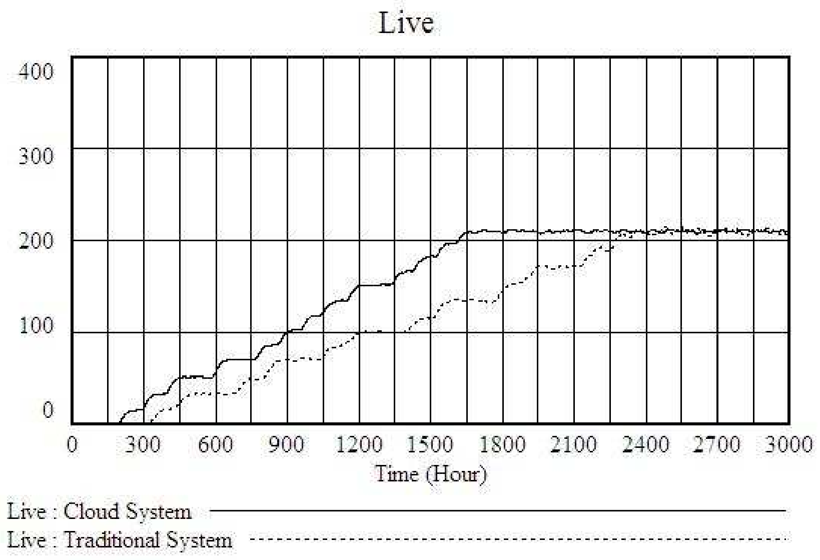


Figure 8.8: Live.

that several aspects can be taken into account to model companies with different needs, and so the conclusions could be different than those obtained by us.

8.4 Summary.

The system dynamic simulation model described in the previous sections has been proposed for highlighting how a Global Software Development environment on the Cloud Platform may facilitate Global Software Development with respect to an environment set up on-premise, using an agile development method. Our aim is to propose a simple tool that every small or medium enterprise can customize and use, in order to verify if a development environment for Global Software on Cloud System could reduce the costs and the time compared to a development environment set up on-premise. Therefore, the results reported in this work are just a simple example to show the potentialities of our model.

We developed a simplified system dynamics model to describe the behaviour of two different approaches to the Global Software Development under similar starting conditions. The development environments analyzed are very complex, and for this reason, in order to avoid a too complex model, we included in our simulator only the variables that in our opinion influence mostly the processes. So, our study has been carried out under some limiting assumptions that could threaten its validity. The proposed model needs to be further elaborated and validated, for example by adding new variables or new relationships among factors. However, this is a first model and one of our future aims is to better answer some questions applying it into detailed case study. This will be the subject of our future work, which will include studies to empirically validate the simulation results using data from GSD real projects and carried on using both Cloud and Traditional environments. Unfortunately it is very difficult to find enterprises available to give answers about any questions related to their software process practices (in particular if you ask about their efficiency, productivity, and so on). This is also one of the reasons which we preferred to start our analysis by implementing a specific model whose parameter values were taken from the literature.

Part III

CONCLUSIONS.

Chapter 9

Concluding remarks

In recent years particular attention has been devoted to the study of complexity theory and its application in the economic field. According to this theory the socio-economic systems are complex adaptive systems, consisting of a large number of individuals, whose interactions create unpredictable behaviour.
The approach of the heterogeneous–agent–based models takes into account the diversity in the various market participants or in the system under test explicitly. The resulting models are not linear and this implies that neither their outcomes, nor their operation can be easily modeled and reproduced in a set of equations.
The heterogeneous–agent–based models offer great advantages for students of economics and for those who want to understand the mechanisms of a complex system in detail. They allow us to obtain information on the behaviour and status of the operators, and can explain the interactions of the system and provide better forecasting tools. One of the drawbacks lies in having to carry out long and heavy development work, although the key to success is inherent in as realistic a description as possible of their behaviour, which makes few assumptions. While making such a description, we should try not to reproduce too specific cases, which burden and complicate the model.
Generally, a complex dynamic system is characterized by a set of separate entities, whose interactions produce non-linear effects, which can not be explained by studying each component individually. The presence of non linearity and conditional behaviour, leads to adopt the simulation as an alternative to analytical models in the study of complex dynamic systems. With a simulation model, we are able to represent a system without any difficulty, studying its aggregate behaviour [49]. Unfortunately, the simulation also involves difficulty in the generalization of the results. It does not provide the same quality and information content as an analytical solution. In fact, while an analytical solution provides full information on the system represented, the simulation is only able to provide information on individual instances of a possible future path of the model, which are often determined by the initial conditions.
In this thesis, this approach is applied to the study of the software market, allowing us to highlight results which would otherwise remain obscure.
In the ***first part*** of this thesis, we studied the CRM software market, through the study of the new pricing models, which in recent years are gaining ground.
Nowadays, vendors cannot neglect the customer power increase in software negotiations.

User willingness to pay is decreased, and nowadays enterprise customers know that they cannot have an agile enterprise with traditional software approach. For these reasons pricing models based on periodic payments, in particular Software as a Service, and Open Source Software are strongly gaining ground.
Our models aim to mirror this new tendency in the market, presenting a new and original approach to analyze the software market, in particular the CRM software market. To our knowledge, this is the first time that the software market has been modelled detailing investment and pricing policies of firms and purchase preferences of customers. For this reason, building the model on existing scientific knowledge has not been simple. Lack of experimental data to initialize or validate the simulations clearly limits the validity of our model, and for this reason our future main objective will be to validate the model using real enterprise data.

We proposed a set of equations for modeling enterprise investments, pricing of products, and the purchase preferences of customers, according to our experience. In Chapter 3, we propose a model to study the competition among OP CRM firms, which enter the market at initial time, and OD CRM firms, which enter the market at next time instants. Given the limited quality of the data available, it makes sense to analyse the business trends in an ideal CRM market reproduced by us through the ***Base Run***. In the ***Base Run*** we adopted the parameter values which can be considered broadly correct and in order to justify these values a detailed sensitivity analysis was performed, studying what happens varying the values of the different parameters. In addition, a Monte Carlo analysis was performed to evaluate the robustness of our model.
We studied a market in which the number of firms and customers is smaller than the real one due to computational reasons. In this market the CRM solutions are directed to a mid-market formed from enterprise customers with 100-1000 employees. The simulation results, of course, depend on the chosen values of the parameters and on the chosen model structure. Indeed, there is relatively little empirical evidence available about the strengths of the various mechanisms on hand.
Despite this, the small market reproduced follows the trends of the real one, concerining both the survival of the firms and the features of the products. In fact, the results of the ***Base Run*** emphasize that only few firms are able to get big market shares, while the remaining firms stay in the market with much smaller market shares.
In addition, the features of the products in the market match the real ones. Indeed, the results highlight that the values of the OP CRM solution quality are greater than that of the OD CRM solution quality, in agreement with the fact that OP CRM solutions are more customized than OD CRM solutions [47]. Furthermore, the prices of OP CRM vary between about 300 and 1,000 for the primary products, and between about 80 and 260 for secondary products; whereas, the prices of OD CRM vary between about 180 and 350, showing trends which match those in the real market.
Lack of reliable information, with respect to the parameters and structure limits qualitative conclusions, even if we feel that the chosen parameters and the equations are broadly correct, and allowed us to obtain a sufficiently realistic reproduction.
However, note that the results reported mean only to show the potentialities of our model and to propose a useful predictive model, to forecast market trends, and investment and pricing policies and business winning strategies. More significant results can be obtained calibrating the model with real data about pricing and investment policies.
Finally, in the first part of this thesis, in Chapter 4, we also present a modified version of the

model described in Chapter 3, for analyzing and studying another segment of the CRM market, which is formed by OD vendors offering Customer Relationship Management products with source code availability, and OD vendors offering Customer Relationship Management products without source code availability.

Also in this model we made considerations similar to the previous one supporting them with a sensitivity analysis and a Monte Carlo analysis.

In the ***second part*** of this thesis, the simulation–based approach has been used for describing the behaviour of three different approaches to the software development under similar starting conditions. We compared by means of the simulation techniques a heavy and prescriptive approach, Waterfall, with two agile and less prescriptive process tools, Scrum and Lean-Kanban.

The simulation–based approach used is that of the ***System Dynamics***, an approach introduced in 1961 from Jay Forrester at MIT (Massachusetts Institute of Technology) and which derives from the study of control systems. Among the techniques of simulation it is the one that more similar to mathematical formalism, since it is based on differential equations.

Moreover, it is based on a reduced use of programming languages, enabling great celerity in the design of the models. For modelling the software development process, and in order to avoid a too complex model, all the variables influencing the processes were not included in our simulation, so our study has been carried out under some limiting assumptions, which could threaten its validity.

The model proposed is a first model and can be considered as a valid starting point for further studies. Its aim is to describe some strengths and weaknesses of the three software development methods by modeling their environment with a continuous-time simulation tool. Although Lean-Kanban is well known in software development processes, it has not yet investigated in depth in research works. In our model, the Kanban workflow was managed through an effective control mechanism to limit the work in progress and minimize the lead time. One of the advantages of this approach is that the work is better controlled, so that the effects of errors are kept limited. On the contrary, in the Waterfall case often projects may fail to complete due to the difficulty to correct errors, including errors in requirements.

Starting from this first model we realized another model for highlighting how a Global Software Development environment on the Cloud Platform may facilitate Global Software Development with respect to an environment set up on-premise. We propose a simple tool, which every small or medium enterprise can customize and use, in order to verify if a development environment for Global Software on Cloud System can reduce the costs and the time compared to a development environment set up on-premise.

We developed a simplified system dynamic model to describe the behaviour of two different approaches to the Global Software Development under similar starting conditions.

The proposed model needs to be further elaborated and validated. It will be the subject of our future work, which will include studies to empirically validate the simulation results using data from GSD real projects and carried on using both Cloud and Traditional environments.

Finally, it should be obvious from whole thesis, that more insights in the underlying processes are needed to give more precise and more conclusive results on all the topics covered. This, however, also requires more and better data, and consequently enterprises or research firms available to give answers about the many questions have its roots in the development of the works dealt with in this thesis.

Bibliography

[1] Abhijit Dubey, Dilip Wagle: Delivering software as a service. Web exclusive, May 2007. [cited at p. 10, 30]

[2] P. Aghion, P. Howitt: Endogenous growth theory. Cambridge and London, MIT Press, 1998. [cited at p. 23]

[3] Amit Dumbre, Satha Priya Senthil, Sidharth Subhash Ghag: Practising Agile software development on the Window Azure platform, White Paper, May 2011. [cited at p. 78, 93, 95]

[4] Andrea Bonaccorsi, Cristina Rossi: Why Open Source software can succeed, Research Policy, vol. 32, pages 1243–1258, (2003). [cited at p. 19]

[5] R. J. Barro, X. Sala-i-Martin: Economic Growth, McGraw-Hill, 1995. [cited at p. 23]

[6] Carina Andersson, Lena Karlsson, Josef Nedstam, Martin Höst, Bertil I Nilsson: Understanding Software Processes through System Dynamics Simulation: A Case Study. In: ECBS, Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, pp. 0041, (2002). [cited at p. 79, 92]

[7] Chichakly Karim: Modeling Agile Development: When is it Effective?. Proceedings of the 2007 System Dynamics Conference. Boston, MA. Print. [cited at p. 77]

[8] Chris Bucholtz: CRM Total Cost of Ownership: Fees, Subscriptions and Hidden Costs, CRM Outsiders and SugarSRM (2011). [cited at p. 20, 27, 37, 38, 60]

[9] Chris Bucholtz: Top 10 Open Source CRM Software Systems, http://www.crmsearch.com/open-source-crm.php [cited at p. 61]

[10] CloudOne: Components of Total Cost of Ownership, July, (2010). [cited at p. 33, 37]

[11] Collabnet: Reinforcing Agile Software Development in the Cloud. Why the Cloud is Advantageous for Agile, and for Accelarating its Enterprise-wide Adoption, White Paper, 2011. [cited at p. 93]

[12] S. D. Conte, H. E. Dunsmore, V. Y. Shen: Software Engineering Metrics and Models. Benjamin/Cummings, (1986). [cited at p. 84]

[13] crmforecast.com. [cited at p. 9, 13, 61]

[14] David Hakala: The Top 10 Open-Source CRM Solutions. Focus Expert Network, http://www.focus.com/briefs/top-10-open-source-crm-solutions/, May 26, 2009. [cited at p. 60]

[15] Emam Hossain, Muhammad Ali Babar, Hye-young Paik, June Verner: Risk Identification and Mitigation Processes for Using Scrum in Global Software Development: A conceptual Framework, 2009 IEEE. [cited at p. 78, 92]

[16] Ernan Haruvy, Suresh P. Sethi, Jing Zhou: Open Source Development with a Commercial Complementary Product or Service, Production and Operations Management, vol. 17, Issue 1, pages 29–43, Blackwell Publishing Ltd, January-February (2008). [cited at p. 20, 32, 43, 64]

[17] Federico Etro: The Economic Impact of Cloud Computing on Business Creation, Employment and Output in Europe. An application of the Endogenous Market Structures Approach to a GPT innovation, 2009. [cited at p. 39]

[18] Galen Gruman, Alan S. Morrison, Terril A. Retter: Software Pricing Tends. PriicewaterhouseCoopers, (January 2007). [cited at p. 29, 31]

[19] Jason Williams, Peter Clegg, Emmett Dulaney: Expanding Choice: Moving to Linux and Open Source with Novell Open Enterprise Server, Published by Novell Press, May 6 2005. [cited at p. 62]

[20] Henrik Kniberg and Mattias Skarin. Kanban and Scrum making the most of both. Managing Editor: Diana Plesa. Enterprise software development series. InfoQ. USA. ISBN 978–0–557–13832–6 (2010). [cited at p. 80]

[21] http://www.crmforecast.com. [cited at p. 9, 13, 61]

[22] http://www.adempiere.com. [cited at p. 17]

[23] http://www.concursive.com. [cited at p. 17]

[24] http://www.consona.com. [cited at p. 18]

[25] http://www.crmsearch.com/salesforce-com.php [cited at p. 10, 13, 14, 15, 16, 18, 19, 41]

[26] http://www.destinationcrm.com/Articles/Editorial/Magazine-Features/The-2012-CRM-Market-Leaders-83897.aspx [cited at p. 10, 61]

[27] http://www.salesforcefoundation.org/ [cited at p. 10, 41]

[28] http://www.sap.com. [cited at p. 14]

[29] James P. Womack, Daniel T. Jones, Daniel Roos: The Machine That Changed the World : The Story of Lean Production. Harper Perennial, (November 1991). [cited at p. 80]

[30] Juthasit Rohitratana, Jörn Altmann: Impact of pricing schemes on a market for Software-as-a-Service and perpetual software, 2012. [cited at p. 36]

[31] Jürgen Bitzer and Philipp J. H. Schröder (Editors): The Impact of Entry and Competition by Open Source Software on Innovation Activity. (December 2005). [cited at p. 19]

[32] Kim E. van Oorschot, Kishore Sengputa, Luk N. van Wassenhove: Dynamics of Agile Software Development. Proceedings of the 27th International Conference of the System Dynamics Society, July 26–30, (2009) Albuquerque, New Mexico, USA. [cited at p. 77]

[33] Kong Xiaoying, Liu Li, Lowe David: Modelling an Agile Web Maintenance Process. (2005). [cited at p. 77]

[34] Lihui Lin: Impact of user skills and network effects on the competition between open source and proprietary software. Electronic Commerce Research and Applications, vol. 7, number 1, pages 68–81, (2008). [cited at p. 19]

[35] Luisanna Cocco, Katiuscia Mannaro, Giulio Concas, and Michele Marchesi: Study of the Competition between Proprietary Software Firms and Free/ Libre Open Source Software using a Simulation model, Software Business Lecture Notes in Business Information Processing, Volume 80, Springer 2011, pp 56-69. [cited at p. 4, 12]

[36] Luisanna Cocco, Giulio Concas, Michele Marchesi, and Giuseppe Destefanis: Agent-Based Modelling and Simulation of the Software Market, Including Open Source Vendors, JITM, Journal of Information Technology Management, Volume 24, number 1, March 2013. [cited at p. 4, 12]

[37] Luisanna Cocco, Katiuscia Mannaro, Giulio Concas, and Michele Marchesi: Simulating Kanban and Scrum vs Waterfall with System Dynamics, Agile Processes in Software Engineering and Extreme Programming Lecture Notes in Business Information Processing Volume 77, Springer 2011, pp 117-131. [cited at p. 5]

[38] Luisanna Cocco, Katiuscia Mannaro, Giulio Concas: A Model for Global Software Development with Cloud Platforms, Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on 5-8 Sept. 2012, page(s): 446 - 452. [cited at p. 5]

[39] Luisanna Cocco, Giulio Concas, and Michele Marchesi: Simulation of the Competition among Traditional and On-Demand Software Vendors, Submitted to Simulation, SAGE Journal. [cited at p. 12, 94]

[40] Luisanna Cocco, Giulio Concas, and Michele Marchesi: Simulation of the On Demand Software Market, Including Vendors Offering Source Code Availability, Submitted to International Journal of Modelling and Simulation, ACTAPRESS Journal. [cited at p. 12]

[41] Maha Shaikh and Tony Cornford: Total Cost of Ownership of Open Source Software. A report for the UK Cabinet Office supported by OpenForum Europe, November 2011. [cited at p. 60]

[42] M. Marchesi, K. Mannaro, S. Uras, M. Locci: Distributed Scrum in a Research Project Management Agile Processes. In Software Engineering and Extreme Programming, Lecture Notes in Computer Science, Volume 4536/2007, pp. 240-244, (2007). [cited at p. 80]

[43] Milan Thanawala, Gordon Smith, Jon Dart, Shivanshu Upadhyay: Oracle SaaS Platform: Building On-Demand Applications, (Sep 2008). [cited at p. 10, 30]

[44] Mikko Mustonen: Copyleft–the economics of Linux and other open source software, Information Economics and Policy, 15, Issue 1, p. 99-121, (2003). [cited at p. 19]

[45] Minghui Wu, Hui Yan: Simulation in Software Engineering with System Dynamics: A Case Study. Journal of Software, Vol 4, n. 10, 1127-1135, Dec 2009, (2009). [cited at p. 78]

[46] Nicholas Economides and Evangelos Katsamakas: Two-Sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry, Management Science, v.52 n.7, pages 1057–1071, July (2006). [cited at p. 19]

[47] OSF Global Services: SaaS CRM vs. on-premise CRM: which is the right choice for your business? ,2012 by OSF Global Services. [cited at p. 42, 58, 104]

[48] P. Dasgupta, R. Das, Dynamic pricing with limited competitor information in a multi-agent economy, in: 7th International Conference on Cooperative Information Systems, 2000, pp. 299-310. [cited at p. 35]

[49] P. Terna, R. Boero, M. Morini, M. Sonnessa: Modelli per la complessitá La simulazione ad agenti in economia, Collana Strumenti,2006. [cited at p. 2, 41, 103]

[50] P. Wernick and T. Hall: The impact of using pair programming on system evolution a simulation-based study. In: 20th IEEE International Conference on Software Maintenance, Proceedings, pp. 422–426 (2004). [cited at p. 78]

[51] Raffaele Giordanelli, Carlo Mastroianni, The Cloud Computing Paradigm: Characteristic, Opportunities and Research Issues, RT-ICAR-CS Aprile 2010. [cited at p. 92]

[52] Rishab Aiyer Ghosh: Study on the Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU, UNU-MERIT, (2006). [cited at p. 20, 23, 32, 43, 64]

[53] Ron Miller: Is Open-Source CRM Right for Your Company?, http://www.insidecrm.com/features/opensource-right-company-030309/, 2009. [cited at p. 61]

[54] Sajid Ibrahim Hashmi, Victor Clerc, Maryam Razavian, Christina Manteli, Damiani Andrew Tamburri, Patricia Lago, Elisabetta Di Nitto, Ita Richardson: Using Cloud to Facilitate Global Software Development Challenges, 2011 IEEE. [cited at p. 78, 92, 93]

[55] SalesForce: Agile Development Meets Cloud Computing for Extraordinary Results at Salesforce.com, White Paper, 2008. [cited at p. 93]

[56] SalesForce: The 7 Secrets of SaaS Startup Success, Copyright ©2008, salesforce.com. [cited at p. 10, 30]

[57] Schwaber Ken: Agile Project Management with Scrum, Microsoft Press,Redmond, WA, (2004). [cited at p. 80]

[58] Schwaber Ken: Scrum Development Process, White Paper, (1997). [cited at p. 80]

[59] Scrum Alliance, http://www.scrumalliance.org

[60] SIIa: Software-as-a-Service Executive Council, Software-as-a-Service;A Comprehensive Look at the Total Cost ofOwnership of Software Applications, (2006). [cited at p. 80]

[61] Sonja Lehmann, Dr. Peter Buxmann: Pricing Strategies of Software Vendors, 2009. [cited at p. 37]

[62] Stefan Slowinski, Richard Nguyen, Surendran Panicker: Global Software On Demand. Shifting the balance of power, (17 September 2007). [cited at p. 35]

[63] Stephen Walli, Dave Gynn, and Bruno Von Rotz: The Growth of Open Source Software in Organizations. [cited at p. 10, 30, 39, 57, 61]

[64] SugarCRM: CRM Total Cost of Ownership Comparing Open Source Solutions to proprietary Solutions, (2005). [cited at p. -]

[65] H. Takeuchi and I. Nonaka: The New New Product Development Game, Harvard Business Review, January-February (1986). [cited at p. v, 20, 27, 59, 60, 61, 62, 64]

[66] The Wall Street Journal Online: Open Source Software Gains in Strategic Value. WSJ: February 8, 2011. [cited at p. 80]

[67] Todd P. Michaud: Software-as-a-Service: A Better Approach for Retailers. Deploy your price optimization software using SaaS. [cited at p. -]

[68] URL: http://agilemanifesto.org/ [cited at p. 61]

[69] VersionOne: State of Agile Survey 2009. Available on the web: http://www.versionone.com/pdf/2009_State_of_Agile_Development_Survey_Results. pdf, Access Date: November 2010. [cited at p. 80]

[70] V. B. Misic, H. Gevaert, M. Rennie: Extreme dynamics: towards a system dynamics model of the extreme programming software development process. In: 5th International Workshop on Software Process Simulation and Modeling (ProSim 2004), W11L Workshop - 26th International Conference on Software Engineering (2004/911), Edingburgh, Scotland, UK, 24-25 May 2004, IEE Seminar Digest, Volume 2004, Issue 911, ISBN: 0 86341 426 5, pp 237–242 (2004). [cited at p. 91]

[71] Vidyanand Choudhary: Software as a Service: Implications for Investment in Software Development. The Paul Merage School of Business University of California, Irvine, (2007). [cited at p. 78]

[72] Warren W. Tignor : Agile Project Management. International Conference of the System Dynamics Society, Albuquerque, NM 2009, July 26 –July 30, 2009. [cited at p. -]

[73] Yang Yong and Bosheng Zhou: Evaluating Extreme Programming Effect through System Dynamics Modeling. Proceedings 2009 International Conference on Computational Intelligence and Software Engineering (CiSE 2009), Wuhan, China, pp. 1–4, (2009). [cited at p. 77]

[74] Yankee Group: UnderstandingTotal Cost of Ownership of a Hostedvs. Premises-Based CRM Solution, (2004). [cited at p. 78]

[75] Yash Talreja: Lean Agile Methodologies Accentuate Benefits of Cloud Computing, 2010. [cited at p. 20, 27, 37, 38] [cited at p. 93]

List of Publications Related to the Thesis

Published papers

- Luisanna Cocco, Katiuscia Mannaro, Giulio Concas, and Michele Marchesi: Study of the Competition between Proprietary Software Firms and Free/ Libre Open Source Software using a Simulation model, Software Business Lecture Notes in Business Information Processing, Volume 80, Springer 2011, pp 56-69.
- Luisanna Cocco, Katiuscia Mannaro, Giulio Concas, and Michele Marchesi: Simulating Kanban and Scrum vs Waterfall with System Dynamics, Agile Processes in Software Engineering and Extreme Programming Lecture Notes in Business Information Processing Volume 77, Springer 2011, pp 117-131.
- Luisanna Cocco, Katiuscia Mannaro, Giulio Concas: A Model for Global Software Development with Cloud Platforms, Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on 5-8 Sept. 2012, page(s): 446 - 452.

Journal papers

- Luisanna Cocco, Giulio Concas, Michele Marchesi, and Giuseppe Destefanis: Agent-Based Modelling and Simulation of the Software Market, Including Open Source Vendors. JITM, Journal of Information Technology Management, Volume XXIV, number 1, 2013.
- **Journal Article submitted but no published yet**
  - Luisanna Cocco, Giulio Concas, Michele Marchesi: Simulation of the Competition among Traditional and On-Demand Software Vendors, Submitted to Simulation, SAGE Journal.
  - Luisanna Cocco, Giulio Concas, Michele Marchesi: Simulation of the On Demand Software Market, Including Vendors Offering Source Code Availability, Submitted to International Journal of Modelling and Simulation, ACTAPRES Journal.

Appendices

Appendix A

Extra Data

This section collects data not presented among those presented in chapter 3 and 4.

A.1 Chapter 3 Extra Data

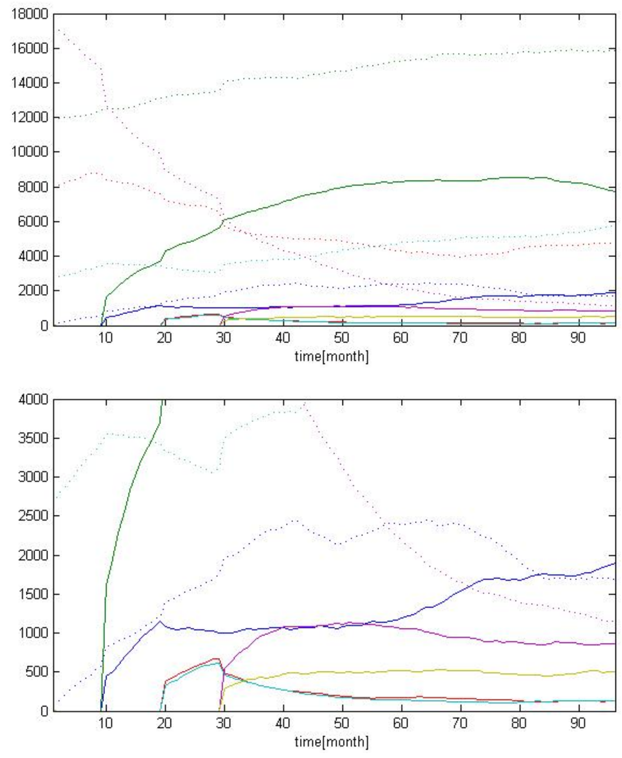


Figure A.1: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 2*). The figure below expands y axis to highlight the customer distributions with smaller values.

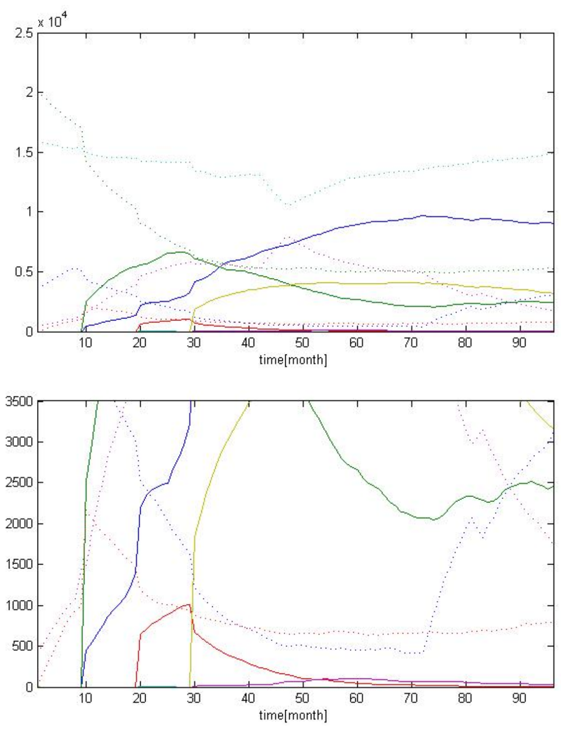


Figure A.2: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 2.1*). The figure below expands y axis to highlight the customer distributions with smaller values.

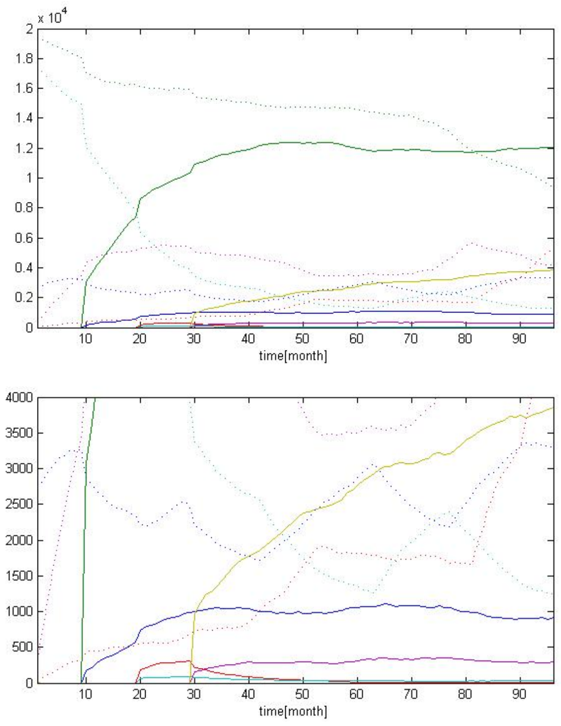


Figure A.3: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (*Set 2.2*). The figure below expands y axis to highlight the customer distributions with smaller values.

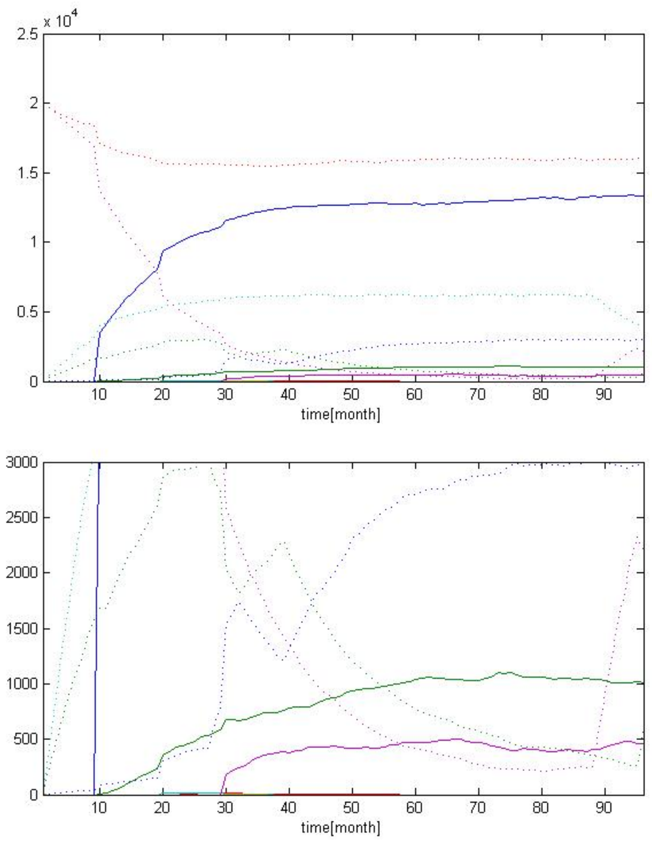


Figure A.4: Customer distributions among OP (dotted lines) and SaaS (solid lines) firms (Set 5.3). The figure below expands y axis to highlight the customer distributions with smaller values.

Table A.1: *Number of customers in the different firms for sets 5, 5.1, and 5.2, and instants t_{entry} and t_{exit} for the firms ousted from the market.*

Firm	*Set 5*		*Set 5.1*		*Set 5.2*	
	OP customers at					
	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$
1	2,282	3,029	20,017	15,845	20,020	75
2	4,751	1,102	19,977	36	5 ($t = 1$)	0 ($t_{exit} = 8$)
3	15,035	340	0	25	19,961	6,868
4	6,291	7,176	0	9	0	5
5	11,641	10,999	6	6,839	14	15,235
	Total OP customers					
	40,000	22,646	40,000	22,754	40,000	22,183
OD customers at						
1	3,107	1,152	64	46	1,319	14,835
2	2,515	2,218	5,446	186	4,329	5
3	2,066	67	1,846	4	1,040	2
4	562	466	15	0	247	3,450
5	881	12,194	1,072	14,713	69 ($t = 30$)	0 ($t_{exit} = 72$)
6	209	1,750	104	2,794	69 ($t = 30$)	0 ($t_{exit} = 76$)
	Total OD customers at					
	9,340	17,847	8.547	17,743	7,073	18,292

Table A.2: *Number of customers in the different firms for Sets 6.i, and instants t_{entry} and t_{exit} for the firms ousted from the market.*

Firm	*Set 6*		*Set 6.1*	
	OP customers at			
	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$
1	5,906	13,397	0	1,041
2	4,201	3,971	20,023	8,615
3	8,207	2,269	0	3,641
4	2,411	1,651	19,972	15,180
5	19,275	3,058	5	1,918
	Total OP customers			
	40,000	24,346	40,000	30,395
OD customers at				
1	837	3562	2,003	6,356
2	2,819	9,332	285	935
3	374	602	918	6
4	533	78	7 ($t = 20$)	0 ($t_{exit} = 27$)
5	285	668	1,106	2,774
6	1,383	1,892	6 ($t = 30$)	0 ($t_{exit} = 37$)
	Total OD customers at			
	6,231	16,134	4,325	10,071

Table A.3: *Number of customers in the different firms for sets 7, 7.1, and 7.2, and instants t_{entry} and t_{exit} for the firms ousted from the market.*

Firm	*Set 7*		*Set 7.1*		*Set 7.2*	
	OP customers at					
	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$
1	1096	400	1	10,015	0 ($t = 1$)	0 ($t_{exit} = 8$)
2	133	384	0 ($t = 1$)	0 ($t_{exit} = 8$)	52	15,715
3	283	3033	1 ($t = 1$)	0 ($t_{exit} = 8$)	0	568
4	292	350	20,159	12,298	0 ($t = 1$)	0 ($t_{exit} = 8$)
5	188	407	0 ($t = 1$)	0 ($t_{exit} = 8$)	19,911	6,281
6	416	385	19,839	2,023	0 ($t = 1$)	0 ($t_{exit} = 8$)
7	10479	9848	0 ($t = 1$)	0 ($t_{exit} = 8$)	20,036	4,957
8	274	407	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
9	18605	4649	0	220	1 ($t = 1$)	0 ($t_{exit} = 8$)
10	8234	8388	0 ($t = 1$)	0 ($t_{exit} = 8$)	(t_{exit} = 8)0 ($t = 1$)	0
	Total OP clients at					
	40000	28251	40,000	24,556	40,000	27,521
	OD customers at					
1	1514	4469	12	977	2	330
2	855	2068	4,369	14,974	2,666	11,971
3	622	1862	0 ($t = 10$)	0 ($t_{exit} = 17$)	614	3
4	310	1252	12 ($t = 20$)	0 (t_{exit} = 121)	20 ($t = 20$)	0 (t_{exit} = 156
5	244	157	59 ($t = 20$)	0 (t_{exit} = 139)	706	3
6	456	1752	0 ($t = 20$)	0 ($t_{exit} = 27$)	0 ($t = 20$)	0($t_{exit} = 27$
7	86	188	11 ($t = 30$)	0 ($t_{exit} = 75$)	526	949
8	76	126	0 ($t = 30$)	0 ($t_{exit} = 37$)	0 ($t = 30$)	0($t_{exit} = 37$
9	350	651	312	283	17 ($t = 30$)	0($t_{exit} = 76$
	Total OD customers at					
	4513	12525	4,775	16,234	4,551	13,256

Table A.4: *Number of customers in the different firms for sets 7.1.1, and 7.2.1, and instants t_{entry} and t_{exit} for the firms ousted from the market.*

Firm	*Set 7.1.1*		*Set 7.2.1*	
	OP customers at			
	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$
1	19	893	0 ($t = 1$)	0 ($t_{exit} = 8$)
2	19,983	1,191	1	2,284
3	0 ($t = 1$)	0 ($t_{exit} = 8$)	0	5,421
4	0 ($t = 1$)	0 ($t_{exit} = 8$)	19,772	9
5	19,996	16,064	306	1,455
6	0 ($t = 1$)	0 ($t_{exit} = 8$)	19,921	15,488
7	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
8	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
9	0 ($t = 1$)	0 ($t_{exit} = 8$)	0	211
10	2	6,285	0 ($t = 1$)	0 ($t_{exit} = 8$)
	Total OP customers at			
	22,000	24,433	40,000	24,868
	OD customers at			
1	0 ($t = 10$)	0 ($t_{exit} = 17$)	0 ($t = 10$)	0 ($t_{exit} = 17$)
2	5,273	15,116	32	1,034
3	24	961	5,030	14,446
4	0 ($t = 20$)	0 ($t_{exit} = 27$)	5 ($t = 20$)	0 ($t_{exit} = 33$)
5	12 ($t = 20$)	0 ($t_{exit} = 74$)	5 ($t = 20$)	0 ($t_{exit} = 49$)
6	8 ($t = 20$)	0 ($t_{exit} = 34$)	12 ($t = 20$)	0 ($t_{exit} = 44$)
7	0 ($t = 30$)	0 ($t_{exit} = 37$)	21 ($t = 30$)	0 ($t_{exit} = 79$)
8	14 ($t = 30$)	0 ($t_{exit} = 61$)	296	379
9	135	261	0 ($t = 30$)	0 ($t_{exit} = 37$)
	OD customers at			
	5,466	16,338	5,401	15,859

Table A.5: *Number of customers in the different firms for sets 8, 8.1, and 8.2, and instants* t_{entry} *and* t_{exit} *for the firms ousted from the market.*

Firm	*Set 8*		*Set 8.1*		*Set 8.2*	
	OP customers at					
	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$	$t = t_{entry}$	$t = T$
1	649	226	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
2	114	259	3	3,159	3	231
3	3104	3503	0 ($t = 1$)	0 ($t_{exit} = 8$)	10,026	7,791
4	900	226	9,907	8,044	9,970	1,298
5	916	197	0 ($t = 1$)	0 ($t_{exit} = 8$)	0	3,434
6	666	820	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
7	189	353	0	172	0 ($t = 1$)	0 ($t_{exit} = 8$)
8	2087	318	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
9	5820	6273	10,090	1,318	1	3
10	5555	512	0 ($t = 1$)	0 ($t_{exit} = 8$)	0 ($t = 1$)	0 ($t_{exit} = 8$)
	Total OP customers at					
	20000	12687	20,000	12,693	20,000	12,757
	OD customers at					
1	260	706	2,111	6,935	3	534
2	1887	5156	1 ($t = 10$)	0 ($t_{exit} = 17$)	0 ($t = 10$)	0 ($t_{exit} = 17$)
3	82	275	3	635	1,979	6,977
4	26	9	60	32	0 ($t = 20$)	0 ($t_{exit} = 27$)
5	31	10	2 ($t = 20$)	0 ($t_{exit} = 50$)	25 ($t = 20$)	0 ($t_{exit} = 39$)
6	447	1496	0 ($t = 20$)	0 ($t_{exit} = 27$)	4 ($t = 30$)	0 ($t_{exit} = 37$)
7	52	62	0 (t =20)	0 ($t_{exit} = 37$)	0	1
8	46	65	147	151	138	199
9	27	4	21	1	9	1
	Total OD customers at					
	2858	7783	2,345	7,754	2,158	7,712

A.2 Chapter 4 Extra Data

Extra Data about the results of the simulation sets.

Table A.6: *Number of customers of the different firms for sets A, B and C*

Firm	*Set A*		*Set B*		*Set C*	
	CS Customers at					
	t_{entry}	at $t = T$	t_{entry}	at $t = T$	t_{entry}	at $t = T$
1	0	5,547	0	0	0	3,101
2	0	0	19,856	16,384	0	1,709
3	0	0	0	4,195	0	0
4	19,904	15,024	0	0	0	0
5	0	0	0	0	19,801	15,795
	Total Number of CS users at					
	19,904	20,571	19,856	20,579	19,801	20,605
OS Customers at						
1	9	0	0	0	55	16,620
2	0	0	20	0	20,135	3,230
3	22	5,680	7	0	2	0
4	2	0	10	16,000	5	0
5	20,063	14,245	20,197	3,915	2	0
	Total Number of OS users at					
	20,000	19,925	20,234	19,915	20,199	19,850

Table A.7: *Number of customers of the different firms in the market for sets B.1 and B.2.*

Firm	*Set B.1*		*Set B.2*	
	CS Customers at			
	t_{entry}	at $t = T$	t_{entry}	at $t = T$
1	0	1,967	0	0
2	0	0	0	25
3	11,218	11,429	0	0
4	0	0	0	0
5	0	0	0	0
	Total Number of CS users at			
	11,218	13,396	0	25
OS Customers at				
1	0	0	0	0
2	4	2,254	40,000	40,451
3	28,772	24,838	0	0
4	1	0	0	0
5	5	0	0	0
	Total Number of OS users at			
	28,782	27,092	40,000	40,451

Table A.8: *Number of customers of the different firms in the market for sets D, E and F*

Firm	*Set D*		*Set E*		*Set F*	
	CS Customers at					
	t_{entry}	at $t = T$	t_{entry}	at $t = T$	t_{entry}	at $t = T$
1	0	0	0	0	0	0
2	10,055	7,667	10,012	7,339	0	0
3	0	0	0	0	0	892
4	0	0	0	0	0	0
5	1	2,815	0	0	9,897	8,479
6			0	0	0	0
7			0	0	0	0
8			0	0	0	0
9			5	3,043	0	1,001
10			0	0	0	0
	Total Number of CS users at					
	10,056	10,482	10,017	10,382	9,837	10,372
	OS Customers at					
1	9,917	1,448	2	0	1	0
2	3	0	11	55	0	0
3	1	0	9,488	1,629	0	0
4	0	0	4	0	2	0
5	23	8,872	3	0	0	0
6			4	0	10,069	8,563
7			37	8,675	0	0
8			429	23	4	0
9			3	6	26	1,855
10			2	0	1	0
	Total Number of OS users at					
	9,944	10,320	9,983	10,388	10,103	10,418