Modeling the dynamics of software competition to find appropriate openness and pricing strategy

By

Thanujan Ratnarajah

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
In
Industrial and Systems Engineering

January 25th 2008

Blacksburg/Falls Church, VA

Keywords: Software, Open source, Proprietary software, Optimization, Dynamic pricing, Systems Dynamics

Modeling the dynamics of software competition to find appropriate openness and pricing strategy

Thanujan Ratnarajah

ABSTRACT

Software firms can use open source development model combined with proprietary development model to increase their profitability. Open source development models can help software firms create products with better technical features at a lower price. Since open source development is a community based development method the popularity of the software among customers will also increase. Using open source development method with proprietary method will also require firms to sell the product at a lower price. This creates a challenge for the firms to find the optimal price and level of openness to maximize their profit.

Using the systems dynamics methodology, development, employment and customer choice for a typical software firm was captured in a simulation model to understand the dynamics of the software firm in a competitive market and to find the optimal level of openness and price. The model was built based on previous research literature, various software models and from the author's understanding of the software industry.

Our analysis suggests that in a fast evolving market where customers spend less time researching and shopping for a software product (Antivirus market VS Operating Systems market), companies should maintain lower level of openness and higher proprietary type development to increase the Net Present Value of the organization. The software firm could benefit from a higher level of openness in a market where the customers base their purchasing decision on the popularity and compatibility of the software and strong network effects are present (e.g. Business intelligence software).

Acknowledgements

I would like to thank my advisor Dr.Hazhir Rahmandad for guiding me through out this entire research. His motivation and encouragement inspired me to come up with useful ideas to make this a fruitful research project. He supported my research by providing valuable guidance and intellectual insights.

I would also like to express my appreciation to Dr. Konstantinos P. Triantis and Dr. Charles P Koelling for providing useful insights to improve the research and stimulate new ideas.

I am very thankful to my family members and friends for their never ending support. I would like to dedicate all my success to them.

Thanujan Ratnarajah

List of Figures:

List of Tables:

1 Chapter 1: Introduction

1.1 Introduction

The software market is a very unique and competitive market in which many small and large organizations coexist. The ability to build products with less over head expense has helped to increase the number of competitors in any software market. In order for an organization to thrive and be successful in such market competition, it should develop strategies and unique methods to increase their market share and revenue.

The recent success of open source licenses has created a new avenue for software organizations to follow and be successful. Initially software organizations would develop software and sell it to the customer and charge them a license fee for using the product. But recent success of open source projects such as Linux, Netscape and Apache has proved to the industry that there are many ways of making profit in this industry. This has also paved way to many software licensing methods based on the development and business models followed by the organization. For example, BSD license was created for organizations which use open source development method and combine the open source software with proprietary software.

Open source development method has many advantages over the proprietary development methods. Open Source software is developed by developers who involve themselves in such project based on their personal interests and to gain knowledge. The open source development method unites many developers with various technical backgrounds and expertise. Hence this approach will bring more variety of talent and creativity to the process at a very low cost or no cost. Having talented and creative developers will help increase the functionality of the software. Also when the community itself is involved in developing the software based on their personal interest it would be convenient for the organization to create a good impression about their product among customers. Sometimes the developers themselves become customers of such products. Using open source development will also require the organization to sell the product at a lower price. No one will want to pay higher price for software that was developed under open source method since the community has the access to the code. This creates a great challenge to the organizations which uses open source code among their software product. They need to find the optimum level of openness and base the price of the software built under hybrid model (open source code mixed with proprietary code) to create larger revenue for the organization.

1.2 Research Objective

Purpose of this research is to understand the dynamics of the competition within the software market to find the optimal level of openness and price under different market conditions such as fast evolving market (Antivirus market VS Operating Systems market) and market with strong customer network effects (e.g.: Business Intelligence software market), to maximize the profit of the software company. Development, employment and network externality structures will be captured in quantitative model to understand the behavior of the software organization and the market under different levels of openness and to optimize the revenue. System dynamics will be used to build the model in this case.

1.3 Problem Definition

The recent success of open source software products such as Linux has brought out the positive aspects of the open source business model. The main factor behind the success of open source is the community based development process. Many developers from various technical backgrounds and expertise contribute to the open source development process. These contributors involve themselves in such open source projects for learning purposes and to undertake new challenges rather than for the monetary benefits. This motivation driven open source community can help bring creativity into software projects.

Commercial software organizations can incorporate open source with proprietary models to create hybrid models which could adopt the positive characteristics of both open source and proprietary models. When open source is incorporated into hybrid models, the software organization could use the motivation driven community based development process to create software with more feature richness at a lower cost. The community based development process also helps to create a larger network externality for the software product creating possibilities for larger market share. But when software is developed under such conditions it will have less value in terms of unit selling price since customers will not pay normal rate that they pay for proprietary software for software which was developed by the community for free. So for an organization to use a hybrid model to its optimum potential to create larger revenue they should decide on the appropriate level of openness and unit price. It is not an easy task for decision makers to decide on the optimal level of openness and price since they need to understand the complete dynamics of the software business clearly before such decisions could be made. Purpose of this research is to understand the dynamics of the competition within the software market to find the optimal level of openness and price under different market conditions such as fast evolving market (Antivirus market VS Operating Systems market) and market with strong customer network effects (e.g.: Business Intelligence software market), to maximize the profit of the software company.

1.4 Research Motivation

There have been much researches done in the past to compare the open source method with proprietary method based on individual components such as software development, network effects on customers, developers, and price. But no research has been done on incorporating all of these components to capture the entire software organization to compare the open source method to the proprietary method. Further it will be more interesting to understand the how these characteristics affect the performance of the organization when open source method is mixed with proprietary method to form a hybrid method under different market conditions.

1.5 Overview of Methodology

Systems Dynamics methodology will be used to develop a quantitative model to analyze the problem in this research. The following steps are used within System Dynamics methodology to develop the model and analyze it.

1.5.1 Problem Articulation

Initial step of this methodology is to define the problem. Important variables in the problem are defined and appropriate time horizon for the model is chosen. The variables for this problem are developed from previous research literature, software models and from the author's understanding of the software industry.

1.5.2 Formulation of Simulation Model

After the variables for the model are defined a qualitative model will be built to represent the problem. Qualitative model will be used to develop a quantitative model using Vensim software.

1.5.3 Verification and Analysis

Once the simulation model is built sensitivity analysis will be performed to understand the dynamics of the systems. During sensitivity analysis the numerical values of variables that affect the model are changed to gain understanding of how the model is affected by these variables.

1.5.4 Policy Design and Evaluation

The sensitivity analysis will help design policies for software organizations. From the sensitivity analysis we will be able understand how an organization could increase its revenue through different policies.

1.6 Organization of Thesis

This thesis is organized into five main chapters. Chapter one provides introduction to the problem which is being analyzed through out the thesis. Chapter two concentrates on literature review in which past research findings are discussed. The model is described in Chapter three. Chapter four presents the results of the analysis and optimization problem. In Chapter five we discuss policies which would help software organization increase its revenue. Future research related to this project will also be discussed.

2 Chapter 2: Literature Review

2.1 Development

The recent success of many open source projects has proven that the open source method could be successfully used to generate revenue in a commercial firm. The main advantage in using open source development method is the access to large number of technical resources (Kenwood 2001). Developers from different technical and cultural background participate in open source projects. The main motivation for these developers to partake is the challenge created by the project. Unlike the proprietary development method, open source development is fueled by motivation rather than monetary benefits (Fuggetta 2003). Developers work on different parts of open source projects based on motivation and their personal technical needs (Dinh-Trong and Bieman 2005). Hence they able to create software product with better functionality. The quality of the technical work produced by these developers is comparable or sometimes superior to the quality of the proprietary development method (Kuan, 2001). The continuous improvement to the source code made by developers in the open source development method also boosts the functionality of the end product (Godfrey and Tu 2000). The community based development process also helps the software firm get feedback about the software product (Lussier 2004).

The community based development in the open source development method creates a negative impact on such software projects. Managing such open source project will become an enormous task when large number of developers works on software development (Madanmohan and De 2004). The software features are updated frequently by the open source developers based on their needs. This also creates a compatibility issue (Cooke et al. 1999). In the current software market most of the products are made to be compatible with different kinds of operating systems. In an open source project the developers make changes to the software based on their needs. This will lead to developers making changes to the software for particular operating systems that they are using. Hence some features will create technical issues when the software is used on different operating systems (Perkins 1999). Proprietary development method has constrained project boundaries which makes it easier to manage compared to the open source development method. The developers in a proprietary environment will have a better understanding of the complete software product. But in an open source development environment developers work on different parts of the software and will not have clear understanding of the entire software (Wilson 1999). The lack of understanding of the entire software will negatively impact the quality of the user help functions of the software.

2.2 Market Share

One of the most influential factors in any organization is the market share of that organization. There have been many published research on market adoption and diffusion models. The market share of a software product depends on individual adopters and decisions made as an organization. The customer's need, compatibility of the product with other software products, feature advantage of the software over the competitor's products, and the total cost of adoption are some of the features that affect the adoption decision process (Higa et al. 1997).

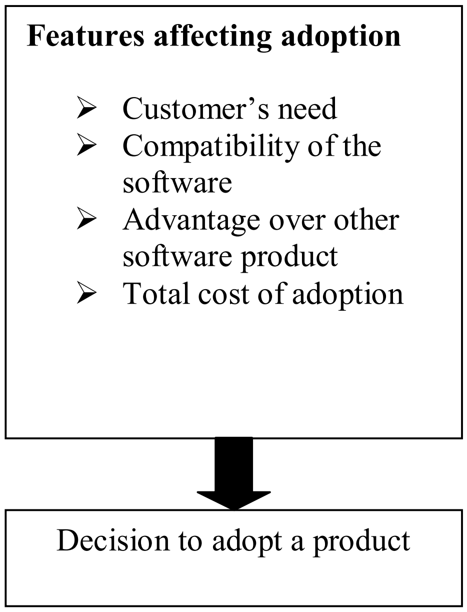

Features affecting adoption
Customer's need
Compatibility of the software
Advantage over other software product
Total cost of adoption
Decision to adopt a product

Figure 2-1: Adoption decision process

The price and user friendly aspect of the software plays a major role is deciding the market share of the organization. When firms decide on purchasing a software product the main feature affecting the decision process is compatibility(Dedrick and West 2004). The software purchased should be compatible with other software products used within the firm. Managing documents will be a tough task to handle if the software products are incompatible (Westarp and Wendt 2000).

Firms looking to buy software would like use a trial version before they actually purchase the product. Software firms with trial versions tend to be popular among consumers since the customers can try out the product before making the purchase (Rogers, 1983). The cost of the trial period should be minimal to attract customers. The cost during trial period also includes the cost of the hardware and technical professionals needed to install and use the product (Khalak, 2001). Software product with good technical help and user interface will be more attractive to the customers (Bonaccorsi, Giannangeli and Rossi 2006).

2.3 Open Source Implementation case study

A case study was performed by Gurbani, Gervert, and Herbsleb to analyze how open source development could affect an organization which incorporates open source method in their process. They chose a “telecommunication-signaling server” software project within Lucent technologies to demonstrate the impact of the open source development method. Initially the software was developed by the designated project group and later was released to all the developers within the organization. The results of their research showed that the number of actively participating developers increased exponentially. Many employees showed interest in providing feedback to improve the software. These feedbacks resulted in increasing the number of new releases within the organization. Exponential increase in the number of new releases helped the organization to create software with better functionally (Gurbani, Gervert, & Herbsleb, 2006).

3 Chapter 3: The Model

This section describes the steps taken to build the model to represent problem under analysis using Systems Dynamics methodology. The first step of developing a model is to define the problem under consideration and define the main variables that affect the problem. The time frame in which the problem will be analyzed is also defined in this section. Based on these definitions a qualitative model is built to help define the problem further. This qualitative model is then used to build the quantitative model.

3.1 The Problem

The recent success of open source software products such as Linux has brought out the positive aspects of the open source business model. The main factor behind the success of open source is the community based development process. Many developers from various technical background and expertise contribute to the open source development process. These contributors involve themselves in open source projects for learning purposes and to undertake new challenges rather than for the monetary benefits (Fuggetta, 2003). This motivation driven open source community can help bring creativity into software projects. Commercial software organizations can incorporate open source with proprietary models to create hybrid models which could adopt the positive characteristics of both open source and proprietary models to create a product with high feature richness at a lower price. When open source is incorporated into hybrid models, the software organization could use the motivation-driven community-based development process to create software with more feature richness at a lower cost. The community based development process also helps to create a larger network externality for the software product creating possibilities for larger market share. Network externality can occur in two ways. The current customers who are using the product will recommend that product to other potential consumers influencing their purchase decision. The second network externality impact will be created when large number of consumers use the product that product becomes the standard of the market. Hence more customers will want to purchase that unit. For example the large number of users in Facebook has attracted more customers to use that for various purposes. Many other applications developed by other developers are also made to be compatible with Facebook platform due to their large installed base. The developers involved in open source projects will also help the organization spread good word-of-mouth about the product to other developers and customers. But when software is developed under such conditions it will have less value in terms of unit selling price since customers will not pay normal rate that they pay for proprietary software for software which was developed by the community for free. So for an organization to use a hybrid model to its optimum potential to create larger revenue they should decide on the appropriate level of openness and unit price. It is not an easy task for decision makers to decide on the optimal level of openness and price since they need not only to understand the dynamics of the software business clearly, but also to deduce the best policies from such understanding. Purpose of this research is to understand the dynamics of the competition within the software market to find the optimal level of openness and price under different market conditions such as fast evolving market (Antivirus market VS Operating Systems market) and market with strong customer network effects (e.g.: Business Intelligence software market), to maximize the profit of the software company.

3.2 Reference Modes

Reference modes are developed to understand how the problem affects important variables over the time horizon. This would help people understand the long term behavior of the variable and its implications (Sterman, 2000).

3.2.1 Installed base of the Software:

The installed base of the software is the total number of units of the particular software that are being used by customers in the entire market. The installed base will follow a bell shaped progress. Initially the software product is adopted by few customers and will take few months for the word of mouth from existing customers to spread and impact the potential customers. As time progresses the demand decreases with decreasing size of overall market size.

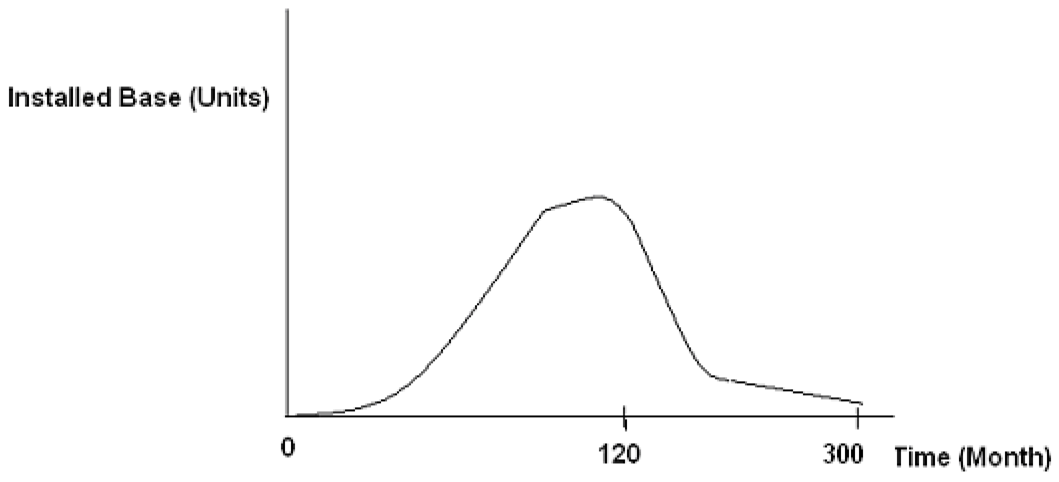


Figure 3-1: Installed Base reference Mode

3.2.2 Net Present Value:

The net present value is defined as the present value of the revenue of the project. The mathematical definition of this variable is:

$$Net\,PresentValue = \sum_{t=0}^{n} \frac{CFt}{(1+r)^{t}} \qquad \textbf{...Equation 3-1}$$

CFt: Cash Flow at time t

r: Discount Rate

n: Total time

Initially the sales rate of the software is low and will not create enough revenue to cover the cost. But with time revenue will grow to cover cost and produce profit.

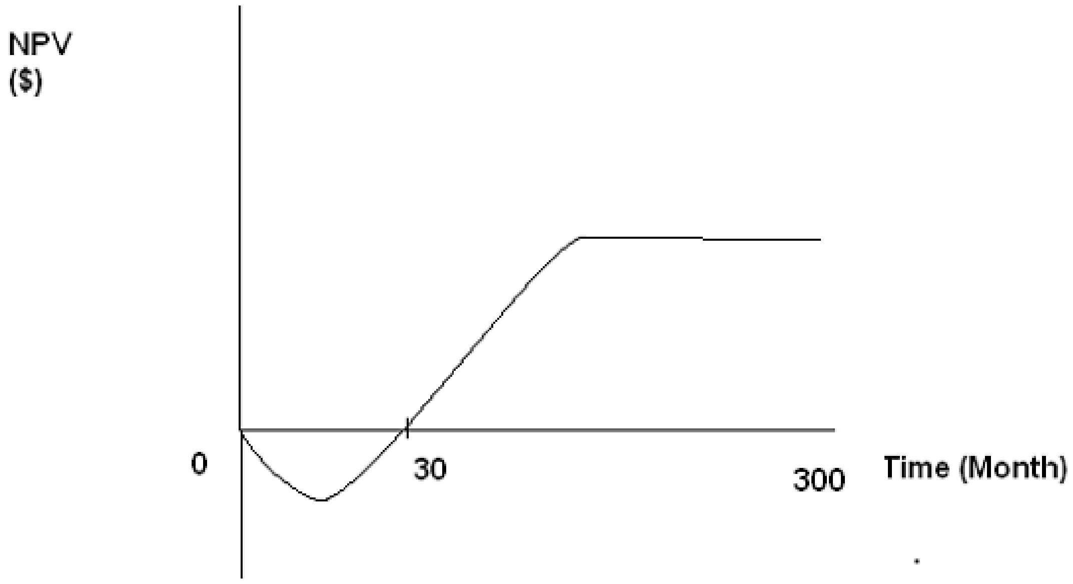


Figure 3-2: Installed Base NPV

3.3 Time Horizon

Time horizon should be defined to cover the complete dynamics of the problem addressed. This should be long enough to cover the dynamics of the system with all the delays (Sterman, 2000). The average life time of the software project is considered to be 150 months in the problem. So Time Horizon is defined to be 300 months from the beginning.

3.4 Assumptions of the Model

The assumptions of the model help the systems to be bound within defined boundaries without affecting the defined problem.

- This problem is defined and modeled for a software market with all the competing companies within the market are new to the market. None of the companies have or had any prior software products in any markets. Including companies with prior product experience in the market will create dynamics to impact the customers purchase decision based on previous product performance. Hence it will be harder to analyze the dynamics of the software competition for a certain product. The boundary of the systems is defined to capture the impact of the current software product in the market.
- The software organizations within this market have the ability to increase their number of employees according to their needs without any budget constraints. However the cost will impact the new present value. Organization will make their

decisions on the success of the project based on the net present value. Hence organizations will try to keep the employment cost to a lower level.

- Revenue for the organization comes from license sales only. The system tries to capture the market competition dynamics of the product based organizations. The project concentrates on software license models rather than business models. Including the business models to cover different ways of making profit will add another separate dimension to the problem. Hence the business models are not included in the model.

3.5 Key Variables

Endogenous variables in this model are variables that are within the software organization structure which affects the dynamics of the software organization performance. Exogenous variables are external variables (outside the software organization systems) which affect the dynamics of the software organization performance (Sterman, 2000). (Refer Appendix1 for definition and units of the variables)

Endogenous	Exogenous
Open Source Fraction	Potential Market Size
Price Fraction	Adopted Market Size
New Features Development rate	Open source Resource
Sales Rate of Software	Employment Cost
(Net) Present Value	
Time for development	
Product Feature richness	
Total Cost	
Return Ratio	
Labor adjustment time	
Employee Attrition Rate	
Average New Features in the Market	
Complementary goods	
Proprietary resources	

Table 3-1: Variable Classification

3.6 Causal Loop Diagram

The causal loop diagram helps to understanding the relationship between variables. This section also describes how theses variables fit into the feedback loops in the software market systems.

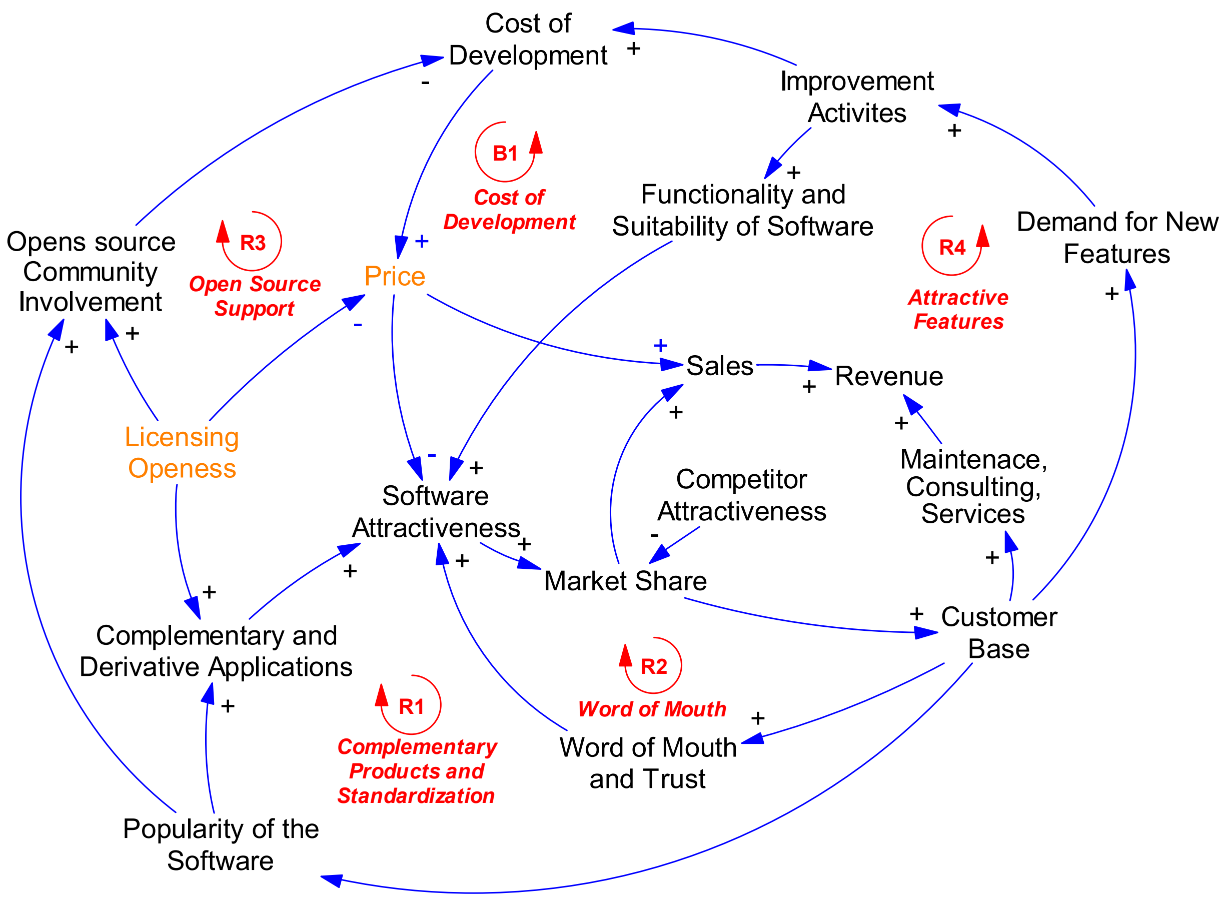


Figure 3-3: Qualitative Model

Complementary Products and Standardization (R1): The increase in software attractiveness will boost the market share and customer base of the product compared to the competitors. When the market share is high the software would be recognized as standard version in the market. Other developers will therefore try to develop their products to be compatible with the focal software. The openness of the software also contributes towards increasing the incentives of others to base their complementary and derivative products on this particular software. Standardization and complementary products increase the attractiveness of the software even further, therefore closing a reinforcing loop.

Word of Mouth (R2): The increase in market share will increase the customer base of the software. Increased satisfied customer base will increase the number of adopters through word of mouth (i.e. satisfied customers will recommend this software to people who they know). Moreover people would trust software that is widely adopted by others. Everything else being equal, cheaper products are more attractive and thus benefit from a stronger word of mouth.

Open Source Support (R3): The involvement of the open source community in a project partly depends on the popularity of the product. On the other hand projects with strong development community can produce better and richer products. Therefore closing a reinforcing loop where open source community reduces the costs and thus increases the attractiveness of the product, which upon higher adoption rates, increases the attractiveness of the product for the open source community. The openness of the software strongly mediates the involvement of open source community with the project and thus the strength of this loop.

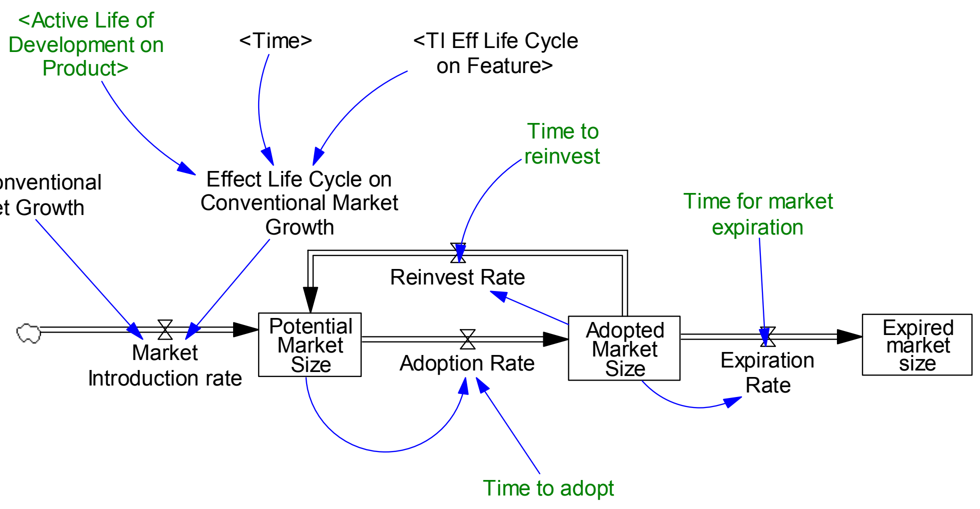
Attractive Features (R4): The increase in the customer base will increase the demand for new features within existing software. Implementing new features will improve the functionality of the software. Hence the software will be more attractive to customers.

Cost of development (B1): Increasing the functionality of the software requires organizational resources. The development costs require high prices that dampen demand, thus checking the potential growth feasible as marginal improvements in the product become more costly or valuable to fewer customers.

3.7 Model Structure

This section explains how the software market, development, network externality, employment, and financial structure of a software organization are captured in the quantitative model. The important variables are defined mathematically within this section.

3.7.1 General Software Market Structure:

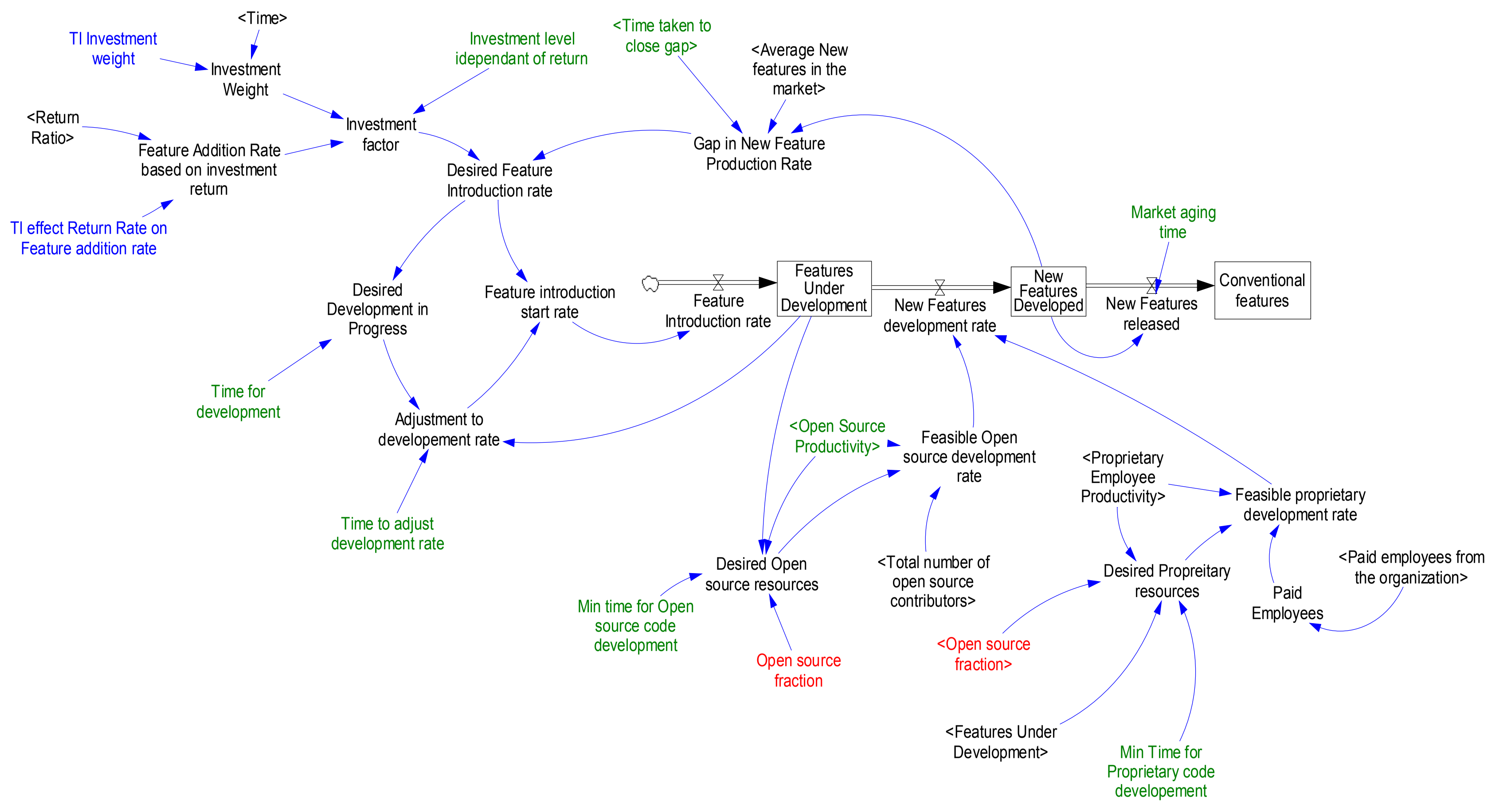


<Active Life of Development on Product>
<Time>
<TI Eff Life Cycle on Feature>
Time to reinvest
Base conventional Market Growth
Effect Life Cycle on Conventional Market Growth
Time for market expiration
Reinvest Rate
Potential Market Size
Adopted Market Size
Expired market size
Market Introduction rate
Adoption Rate
Expiration Rate
Time to adopt

Figure 3-4: General Software market structure

Potential Market size is decided based on the Market Introduction Rate. The consumers within the Potential Market for the particular software then adopt a software to become the Adopted Market Size. After users in the market have adopted one of the software firm's product they stick to that particular software until the Time to Reinvest or until market expiration time. In the initial model Time to reinvest is set to 50 months. Once the decision to reinvest is made, the customers once again become part of Potential Market Size since they have the option of reinvesting in the same software firm's product or in a competing firm's product. After a time period software becomes obsolete and the market expires since customers are not interested in investing in the software. The Expired market size depends on the Time for market expiration..

3.7.2 Software Development:



<Time>
TI Investment weight
Investment Weight
Investment level idependant of return
<Time taken to close gap>
<Average New features in the market>
<Return Ratio>
Feature Addition Rate based on investment return
Investment factor
Desired Feature Introduction rate
Gap in New Feature Production Rate
TI effect Return Rate on Feature addition rate
Market aging time
Desired Development in Progress
Feature introduction start rate
Feature Introduction rate
Features Under Development
New Features development rate
New Features Developed
New Features released
Conventional features
Time for development
Adjustment to developement rate
<Open Source Productivity>
Feasible Open source development rate
<Proprietary Employee Productivity>
Feasible proprietary development rate
Time to adjust development rate
Desired Open source resources
<Total number of open source contributors>
Desired Propreitary resources
<Paid employees from the organization>
Paid Employees
Min time for Open source code development
Open source fraction
<Open source fraction>
<Features Under Development>
Min Time for Proprietary code developement

Figure 3-5: Software Development

The new Feature Introduction rate mainly depends on the demand for new features in the market. The software developing organization decides on the number of features to develop based on the Return Ratio and the Gap in New Features in the market. The return ratio indicates what fraction of the total cost the organization was able to create as its revenue. If the fraction is less than one it indicates that the revenue was less than the total cost (the organization has reported a net loss). For a profitable project the return ratio will be greater than one.

Return Ratio= Total Revenue/Total Cost **...Equation 3-2**

When the Return Ratio is greater than one the organization will want to close the gap in new features between the competing software products. If the Return Ratio is less than one it indicates low demand for the software and the organization will decide to close the gap partially. This decision is made using the following look up table function:

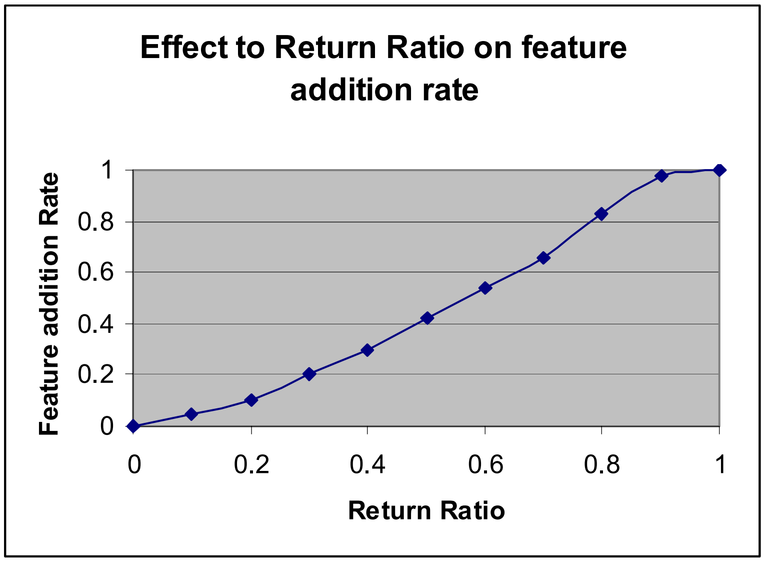


Figure 3-6: Effect of Return Ratio on feature addition rate

The gap in the New Feature production rate is determined by comparing and finding the difference between the average new features in market and the new features in the software introduced by the particular organization. New feature introduction rate also takes into consideration the gap caused by the development delay.

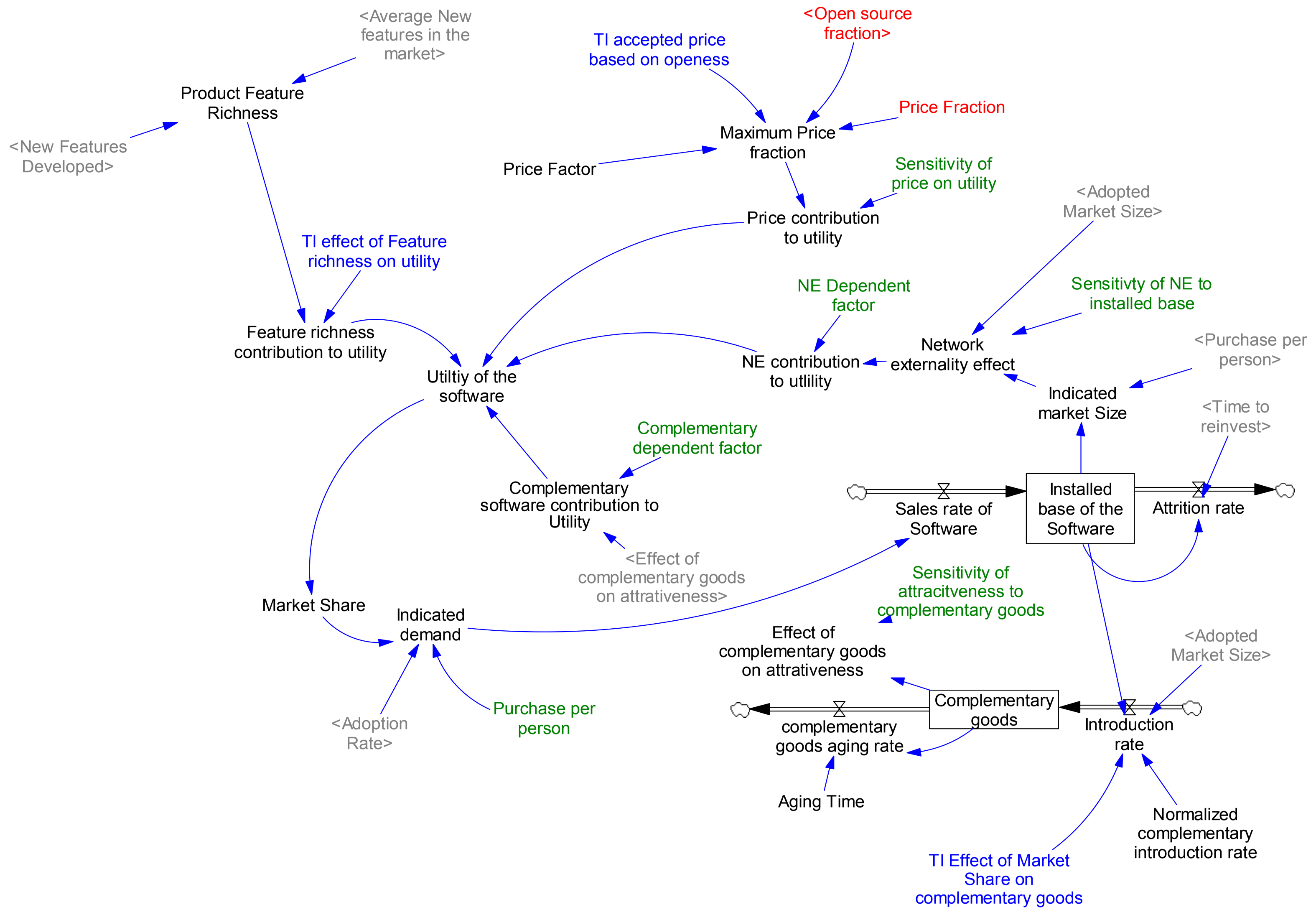
> Gap in New Feature Production Rate=(Average New features in the market-New Features Developed/Time taken to close gap **...Equation 3-3**

The stock, Feature Under Development, converts to New Features Developed based on the New Feature Development Rate. This development rate depends on the proprietary and open source contributors' development rate.

The productivity of the open source contributors will be less than the productivity of the proprietary developers. Open source contributors work on the development process in their spare time and based on their needs. Whereas the proprietary developers paid by the software company work according the organization's schedule to develop the software.

After some time the New features in the market will become Conventional Features. At this point any software in the market will be expected to have those features in them and the features lose their competitive value.

3.7.3 Network Externality:



<Average New features in the market>
Product Feature Richness
<New Features Developed>
TI accepted price based on openess
<Open source fraction>
Price Fraction
Maximum Price fraction
Price Factor
Sensitivity of price on utility
Price contribution to utility
<Adopted Market Size>
TI effect of Feature richness on utility
NE Dependent factor
Sensitivty of NE to installed base
Feature richness contribution to utility
Network externality effect
NE contribution to utility
<Purchase per person>
Utiltiy of the software
Indicated market Size
<Time to reinvest>
Complementary dependent factor
Complementary software contribution to Utility
Sales rate of Software
Installed base of the Software
Attrition rate
<Effect of complementary goods on attrativeness>
Sensitivity of attracitveness to complementary goods
Market Share
Indicated demand
Effect of complementary goods on attrativeness
<Adopted Market Size>
Complementary goods
complementary goods aging rate
Introduction rate
<Adoption Rate>
Purchase per person
Aging Time
Normalized complementary introduction rate
TI Effect of Market Share on complementary goods

Figure 3-7: Network Externality

Utility of the software:

Utility of the software helps define the market share of the organization. The utility of the software is affected by the feature richness, network externality, complementary products, and price. Potential customers in the market will be interested in purchasing software, which has better functionality, popularity, and lower price. Increased functionality and popularity will increase the utility of the product since more people will want to purchase the product. But higher price will drive away potential customers decreasing the value of the product.

Utility of the software= NE contribution to utility+ Feature richness contribution to utility-Price contribution to utility + Complementary software contribution to Utility **...Equation 3-4**

Price Contribution to Utility:

The price of software mainly depends on the Open source Fraction. If an organization decides to open certain percentage of the software to the community they will have to decide the price based on the level they decide to keep open. The model uses the following look up function to determine the price of the software based on the openness:

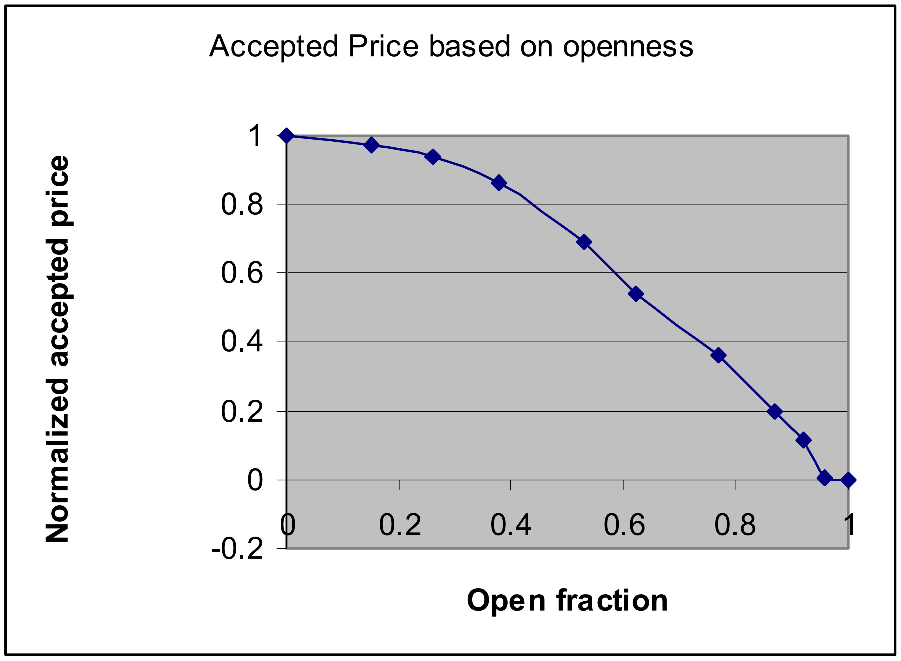


Figure 3-8: Accepted price based on openness

If an organization decides to open a very small percentage to the public it would not affect the price drastically since the developers do not have access to main parts of the code for free. At the same time if that organization decides to open majority of the software code, the price of the software will drop drastically since the developers can modify the open codes to create another free version of the software.

The Management can decide the actual price fraction based on the strategy they decide to follow. Price fraction, a variable between 0 and 1, allows the management to change pricing strategy. The model captures the price contribution to the utility as a function of the Price fraction decided by management, Sensitivity of price to utility and the Price factor. Sensitivity of price to utility and Price factor determines the market sensitiveness to the price of the software.

Price contribution to utility=Price Factor* (Min(Price fraction decided by management[Software]^Sensitivity of price on utility),0)) **...Equation 3-5**

Product Feature richness contribution to utility:

Product feature richness is determined by comparing the New Features developed by the organization and the Average new features in the market. The software will be considered to have a higher feature richness when the ratio of the two variables (the New Features developed by the organization and the Average new features in the market) is greater than one. When the feature richness of the software is less than the average feature richness of its competitor's software its contribution to utility will be minimal since the customers will prefer to buy a software with better operational functions. But after certain level of increase in the feature richness the utility will not be affected since having too many features will result in some features being not used by the common customers. The feature richness contribution to the utility follows a S-shape function.

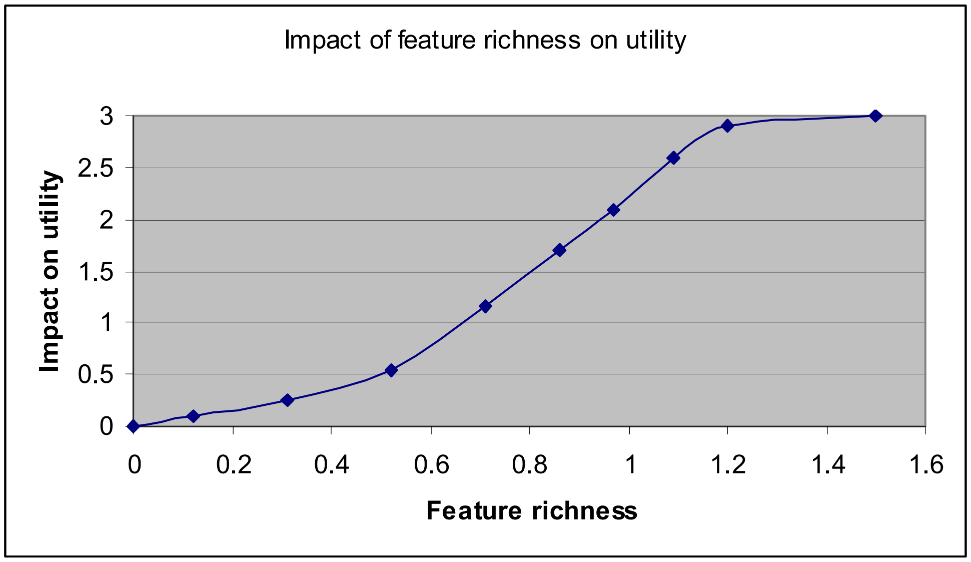


Figure 3-9: Impact of feature richness on utility

Network Externality contribution to utility:

Popularity of a software product has a great impact on its market share. When a software product becomes popular over its competitors' product it becomes the standard product of the market and more potential customers will be motivated to purchase that software. Also the current users will help to increase the popularity of the software through word of mouth. The model captures this dynamics by assessing the popularity of software by comparing the Indicated market Size with the Potential Adopters and finding its ratio. The Indicated market size defines the number of customers using this particular software. This value is determined by the 'Installed base of the software' stock variable. Installed base of the software is the number of units being used in the market by the consumer. This stock variable depends on the Sales rate and Attrition rate. The Sales rate depends on the market share of the organization and the Adopted market size. It is assumed that each customer indicated by the market share will buy one unit at any given time. After every Time to Reinvest (50 months) period the customer will deiced to either reinvest in the same software or a different one. This would drain the installed base since the decision to reinvest would make the customer a potential customer for all competing software organizations in the market.

The Network externality effect is calculated in the model by raising the popularity (ratio of Indicated market Size and Adopted Market Size) to the power of Sensitivity of network externality to installed base. Base on the sensitivity factor the externality effect will increase rapidly with the increase of popularity of the software.

Network externality effect = (Indicated market Size/Adopted Market Size)^Sensitivity of NE to installed base **...Equation 3-6**

NE contribution to utility = NE Dependent factor*Network externality effect **...Equation 3-7**

Effect of Complementary products on utility:

The software's popularity will encourage other kinds of software product developers to create software that are compatible with that particular software. When decisions on purchasing software have to be made by customers, one main factor they would look at is compatibility between popular software products. This could be easily understood in an operating system software market. The most software developers build their software for the most popular operating systems and the customers will purchase the operating systems with the most number of compatible software products. So increasing popularity will increase complementary products and in return complimentary products will increase the popularity of the software even further.

This concept is captured in the model by a similar structure to that of the Network Externality. Complementary goods introduction rate depend on the popularity of the software (ratio of Indicated market Size and Adopted Market Size) and is calculated using the following look up table:

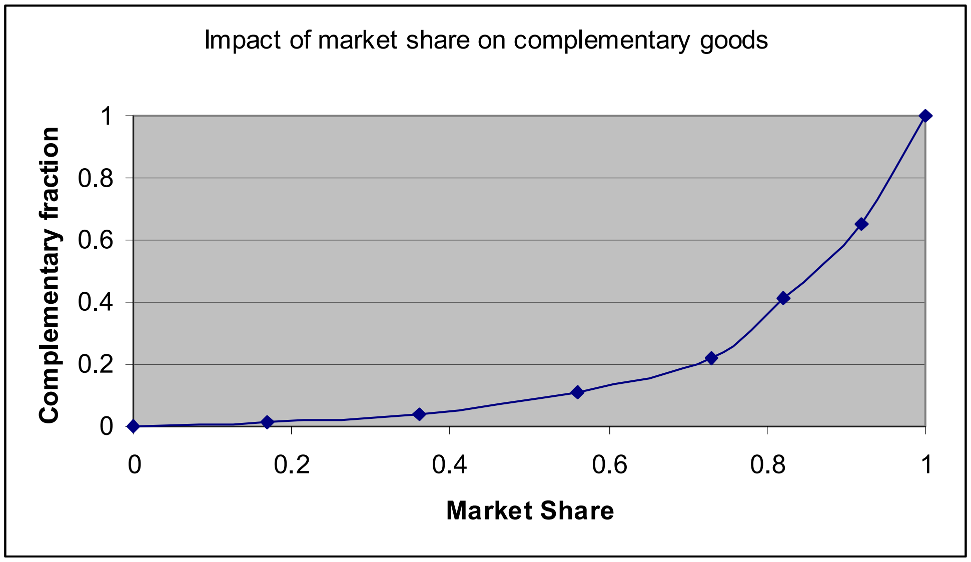


Figure 3-10: Impact of market share on complementary goods

When the ratio of Indicated market Size and Adopted Market Size is low not many developers will be motivated to produce complementary products for the software. But with the increase of the ratio the complementary product introduction rate will increase exponentially.

Market Share:

The market share is formulated as:

$$\mathbf{M1} = \frac{e^{u1}}{\sum_{n=1}^{n=p} e^{un}} \qquad \textbf{...Equation 3-8}$$

- M1: Market share of organization number 1
- P: Number of competing software organizations in the market
- U: Utility of the software

This formulation ensures that the organization with high feature richness, complementary products, and Network externalities would attain higher market share compared to its competitors. The increase in price over its competitors would decrease the market share.

3.7.4 Paid Employee Structure:

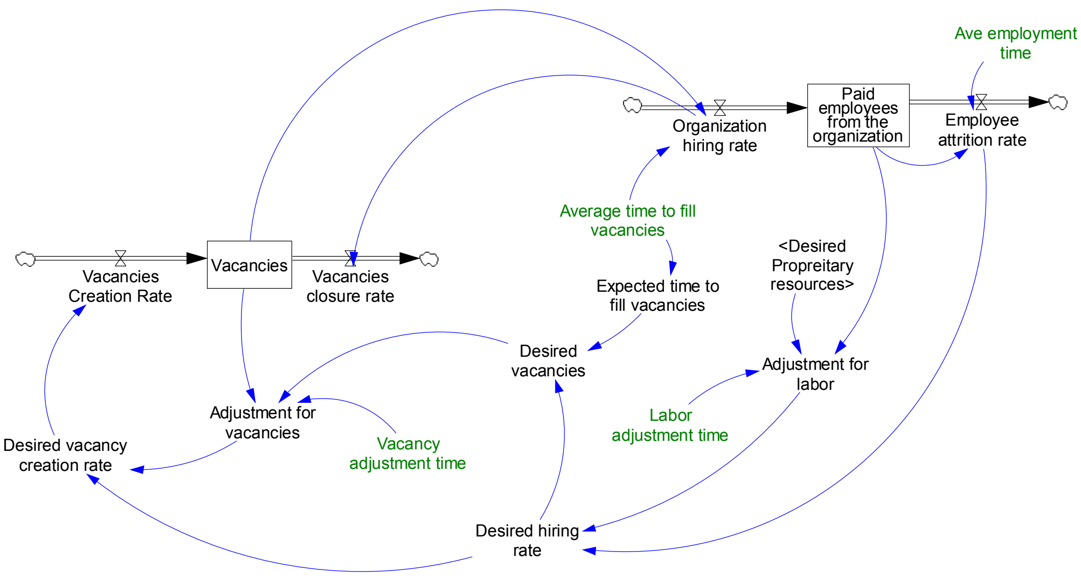


Figure 3-11: Paid employee structure

The total number of paid employees in the organization is represented by a stock and flow structure. Hiring rate increases the paid number of employees while the Employee Attrition rate decreases this stock. The Hiring Rate depends on the Vacancies created and by Time to fill those vacancies. The Employee Attrition Rate includes the employees leaving the organization for other opportunities and retirement. The hiring process is not an instantaneous process. It takes time to determine the number of desired employees, advertise, and hire the right candidate. All these delays are captured by the ‘Vacancies’ stock and flow structure.

Initially the Desired Hiring Rate will be determined by the Desired Proprietary resources and Labor Adjustment time. The Desired vacancy creation rate should take the Desired hiring rate and the Adjustment for vacancies into consideration. The Adjustment for vacancies is created by the hiring process delay in the systems. The graph below shows the behavior of the Desired vacancies and Vacancies. Initially number of Desired vacancy is leading the number of Vacancies due to the fact that organization has realized the need to increase vacancies but they cannot increase it at the same level as Desired vacancy due to the managerial delays. In the latter part when the Desired vacancies decrease it will take time for the organization to adjust the Vacancies accordingly. So when determining the Desired vacancy creation rate the Adjustment for vacancies need to be considered in order for the organization to close the gap in vacancies created by the managerial delay.

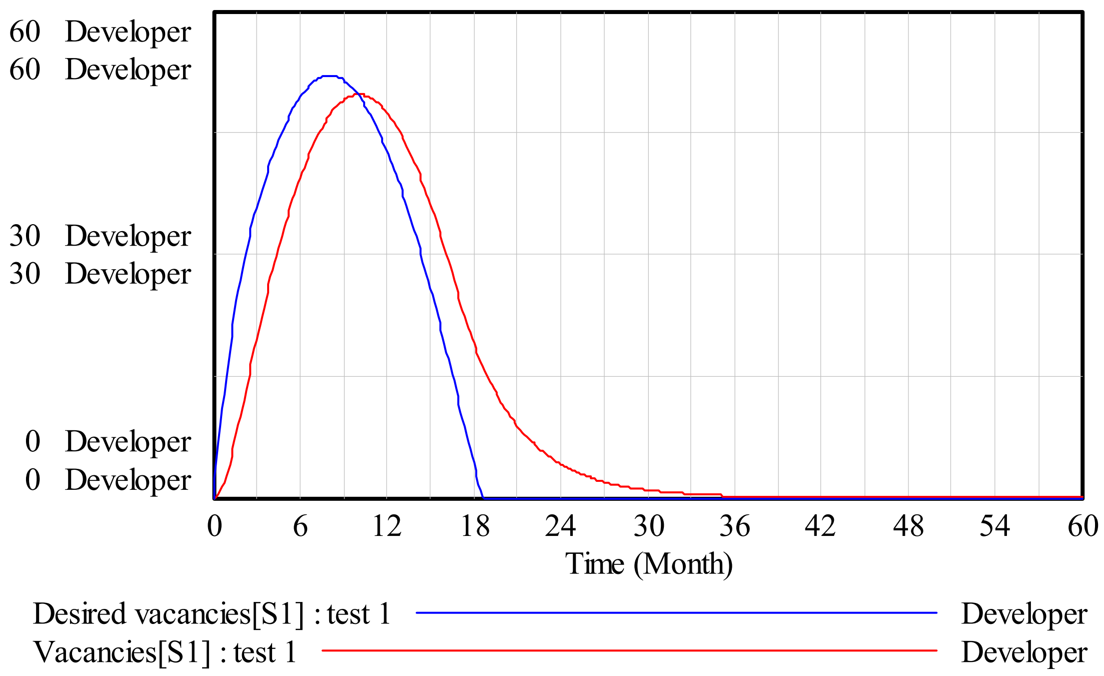


Figure 3-12: Vacancies Dynamics

The organization will close the Vacancy when an employee is hired for that position by the organization.

3.7.5 Open Community Contributors:

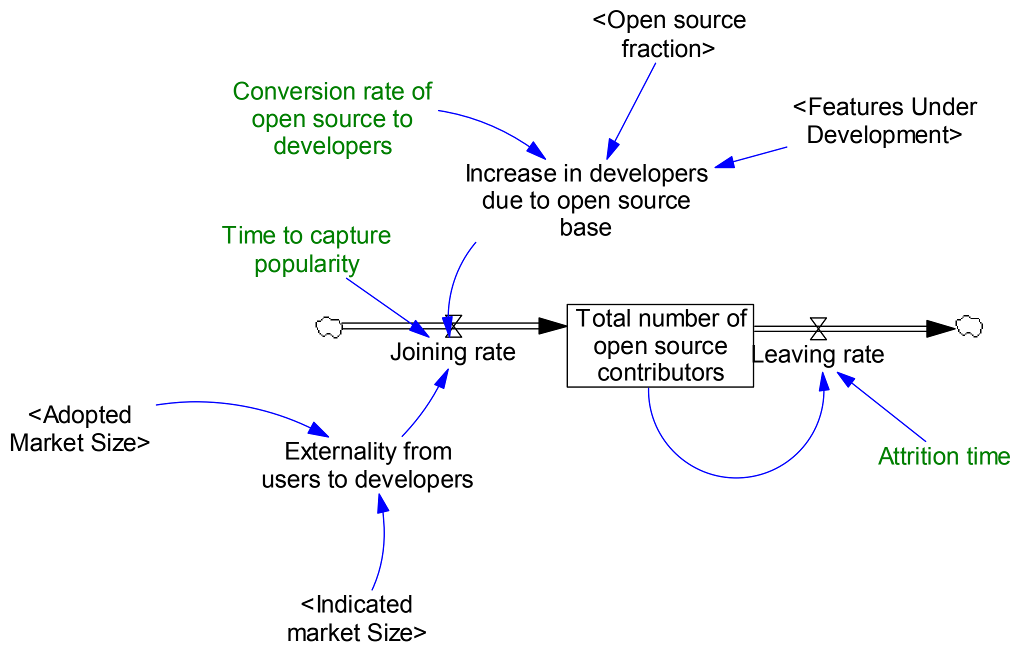

<Open source fraction>
Conversion rate of open source to developers
<Features Under Development>
Increase in developers due to open source base
Time to capture popularity
Joining rate
Total number of open source contributors
Leaving rate
<Adopted Market Size>
Externality from users to developers
Attrition time
<Indicated market Size>

Figure 3-13: Open Source contributor structure

The popularity of the open source projects is the main factor that determines the Total number of open source contributors. The popularity of the software is measured by the number of adopters of that particular software over its competitors. The Externality from users to developers follows an S-shape function with the popularity increase. When the level of popularity is low the open source project will not attract many contributors. At the same time after certain level of increase in the popularity level the level of attraction will not change from its maximum value.

Externality from users to developers[Software]=(1/(1+EXP(-ZIDZ(Indicated market Size[Software], Adopted Market Size)))) **...Equation 3-9**

Increase in developers due to open source base is defined to be the multiplication of number of open source features and the Conversion rate of open source to developers.

Conversion rate of open source to developers defines the number of open source contributors who will be attracted to work on each open source feature (Parker and Van Alstyne 2005).

Increase in developers due to open source base[Software]=Conversion rate of open source to developers*Features Under Development[Software]*Open source fraction[Software] **...Equation 3-10**

Joining rate[Software]=Externality from users to developers[Software]*Increase in developers due to open source base[Software]/Time to capture popularity **...Equation 3-11**

3.7.6 Financial Structure:

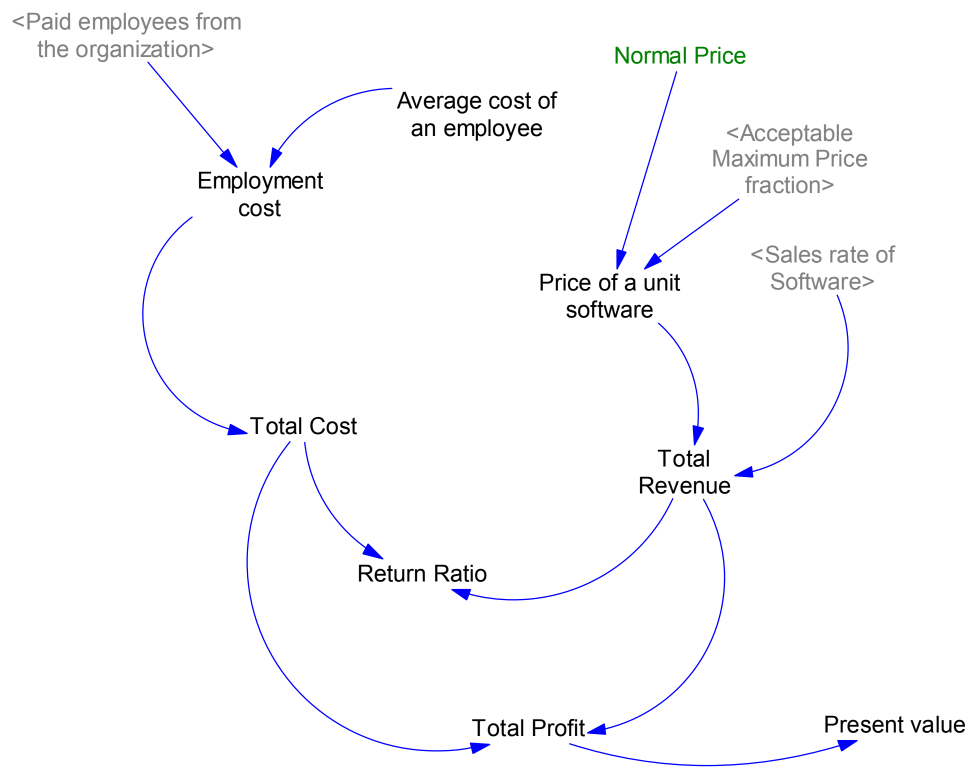


Figure 3-14: Financial Structure

The organization’s only cost comes for labor. This labor cost is calculated by multiplying the total number of paid employees by the average cost of an employee. The revenue of the organization is created through license sales.

Present value is used as the main variable to define the profitability of this project. Net present value is calculated with 8% discount rate.

Present value[Software]=NPV(Total Profit[Software],0.08,0,1)**...Equation 3-12**

Return Ratio[Software]=ZIDZ(Total Revenue[Software], Total Cost[Software])

...Equation 3-13

4 Chapter 4: Results

The results of the simulations from the model described in Chapter 3 will be discussed in this chapter. The sensitivity of model will be analyzed in the first section of this chapter. The sensitivity analysis will be performed by changing some of the important parameters within the model to observe the changes in Net Present Value (NPV) and the market share value. In the second section, the model will be optimized to find the optimal level of openness and price fraction. For the optimization purpose we will use the optimization tool within Vensim.

4.1 Analyzing three firms in competition

In order to test the behavior of the model we perform a numerical analysis on the model. During numerical analysis the values of variables which affect the model the most are increased and decreased by certain percentage. This would help understand the dynamics of the model.

4.1.1 Software Market competitors with same level of openness

The three organizations within the market considered are assumed to have the same level of openness. All three organizations maintain 20% openness of their total features to be developed. In this section we analyze the impact of aggressive development, adoption rate, and the impact of sensitive market on a certain software organization.

4.1.1.1 Base Case

In this market all three competitors will have equal market share due to the similarity in their strategies (similar level of openness). The potential market for these organizations increases at the beginning and starts to decline as time progresses due to the product becoming obsolete. Organizations will stop developing new features around 60 months after all the required new features are added to the software. But the demand of the software will continue for longer timer period since people are interested in buying the software even though they do not upgrade the product. Open source developers working on such projects will leave the project as soon as the organization decides to stop introducing new features to the product.

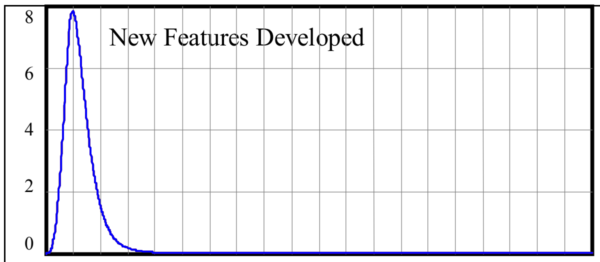
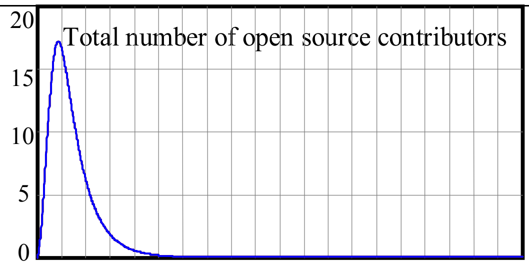
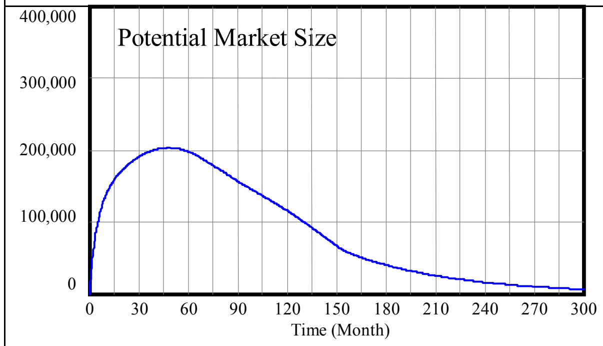
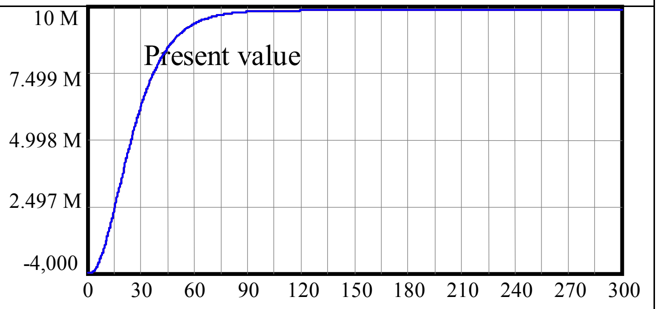
 **Figure 4-1: Same level of openness: Base Case-New Features Developed**	 **Figure 4-2: Same level of openness: Base Case-Open Source contributors**
 **Same level of openness: Base Case-Figure 4-3: Potential Market Size**	 **Same level of openness: Base Case-Figure 4-4: NPV**

Table 4-1: Base Case behavior (S1=S2=S3=0.2)

4.1.1.2 Impact of aggressiveness in feature development

Time taken to close gap is the parameter that determines how fast the organization will want to match up to the market competitors' product's average new features. For example virus scanning software will have to add new features aggressively to prevent new virus attacks where as an Operating Systems software can update the new features much slower than the virus scan. The organization can either be aggressive and increase the speed the process of closing gap or be passive and observe the market's behavior before trying to catch up to the competitors.

In Case 1 the organization decides to be aggressive and shorten the Time to close Gap to 5 months. This aggressiveness will immediately increase the feature introduction rate of the organization. The aggressive new feature production will continue until the feature of the software is equal to the average new features in the competitors' software. The adjustment in development will take time due to the development delays in the system. Hence the organization will start to lead the market in new features developed. As shown in Figure 4-1, the lead in the new features will be start to decrease around 14th month and will reach steady state when there is no gap in new features.

This aggressive approach to close gap in new features will create an increase in new features being developed (Figure 4-2). The increase in new features will increase the Feature richness of the software. Software with better functionality will be more attractive for the customers hoping to purchase the software. Hence the utility of the software will increase due to the feature richness increase which will result in creating higher marker share.

Increased market share will assure higher installed base for the organization. The increase in installed base will create a better name for the software among the consumers causing a larger impact through network externality. At the same time more complementary software products will start to emerge in the market. The positive reaction from complementary goods and network externality will feedback into the system to increase the market share further.

Increasing market share will increase the total revenue of the organization (Figure 4-5). Initially the profit (for about 8.75 months) is less than the base case due to the higher employment cost. Increased new production will require more paid and open source contributors to work on the development process. This increase in employment will increase the cost and the organization will not be able to recover that cost until 8.75 months from the beginning.

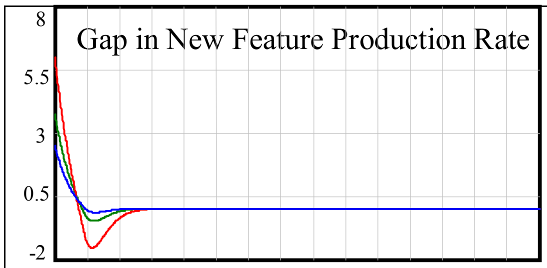


Figure 4-5: Same level of openness: Aggressive development-Gap in new feature production rate

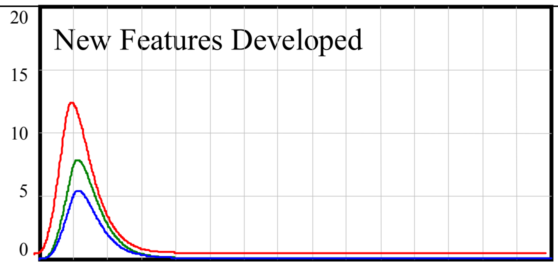


Figure 4-6: Same level of openness: Aggressive development-New Features developed

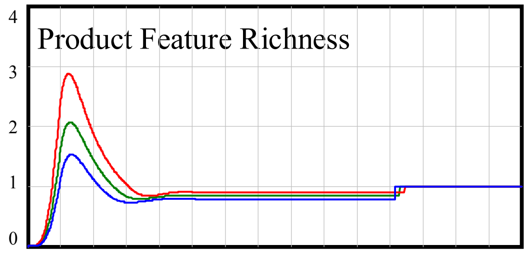


Figure 4-7: Same level of openness: Aggressive development-Product feature richness

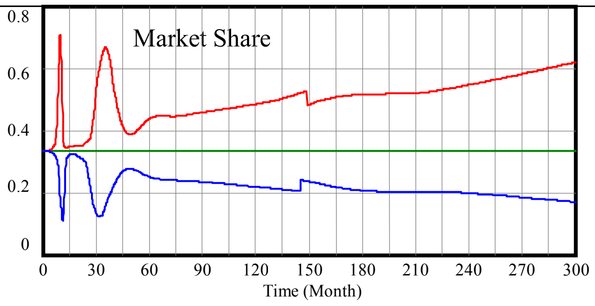


Figure 4-8: Same level of openness: Aggressive development-Market Share

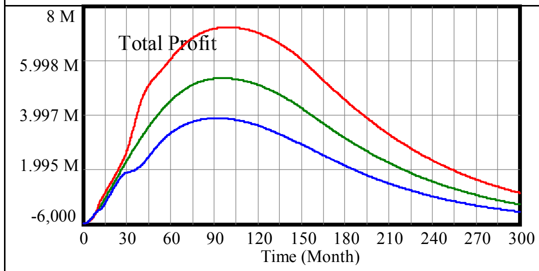


Figure 4-9: Same level of openness: Aggressive development-Total profit

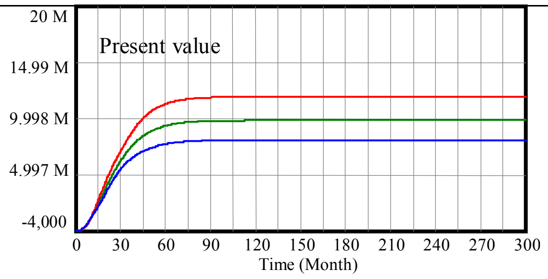


Figure 4-10: Same level of openness: Aggressive development-NPV

Present value [S1]: Closing Gap Slower 🟦

Present value [S1]: Closing Gap Faster 🟥

Present value [S1]: Base Case 🟩

Table 4-2: Aggressive feature development (S1=S2=S3=0.2)

4.1.2 Software Market competitors with different level of openness

In this section the model will contain three subscript values to represent three organizations with different levels of openness. The open source fraction of the three organizations (S1, S2,and S3) are 0, 0.5, and, 0.9 respectively. This section analyze the impact of the aggressive development, adoption rate, and strength of customer network in a competitive market and understand how the competition dynamics affect the entire market.

4.1.2.1 Base Case

The base case analysis is used to identify how the level of openness affects the variables that contribute to the net present value of the organization.

Organization S1 leads the market in new features developed (Figure 4-17). This is mainly due to the higher development rate of that organization. Organization S1 has the least amount of open source features. Most of their development is done by paid developers who are hired by the organization. Process of hiring paid employees is faster than creating network externality to motivate open source contributors to participate in the development process. This is the main reason for Organization S1 to have a higher development rate initially (for about 13 weeks). After establishing a steady base S2 and S3 are able to create larger network externality to increase the number of open source contributors. Once this is done S2 and S3 are able to overtake the development rate of S1.

The market share of S3 will be higher than its competitors. Since S3 has 80% open source features their product will be sold at a lower price in the market. Initially more customers will adopt this product over its competitors because of its low price. Around 8th month organization S3 will lose some of its market share due to the lower feature richness. But will regain the market share once the network externality impact increases to increase the feature richness. S1 will have the lowest market share since the price of the software is higher due to the low open source fraction. This also contributes towards lower network externality impact.

Even with the low market share organization S1 will have higher revenue than its competitors due to higher unit selling price. But the employment cost will also be higher for this organization since most contributors are paid by this organization. Hence the net present value of S2 will be higher than S3 because of lower employment cost.

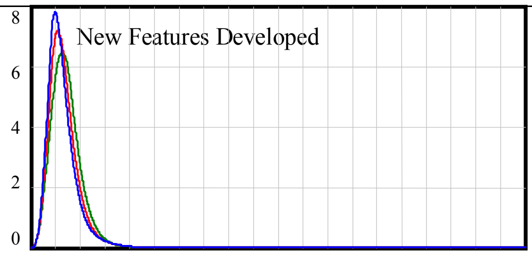


Figure 4-11: Different level of openness: Base Case- New features developed

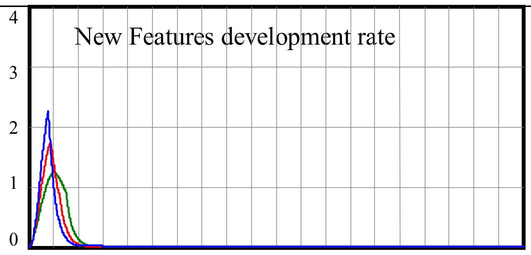


Figure 4-12: Different level of openness: Base Case- New features development rate

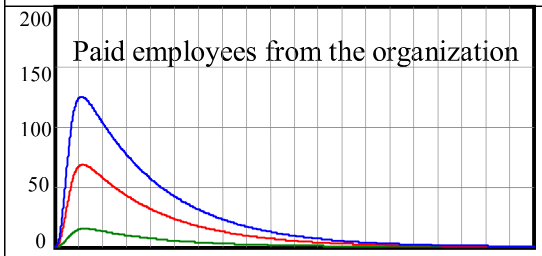


Figure 4-13: Different level of openness: Base Case- Paid employees

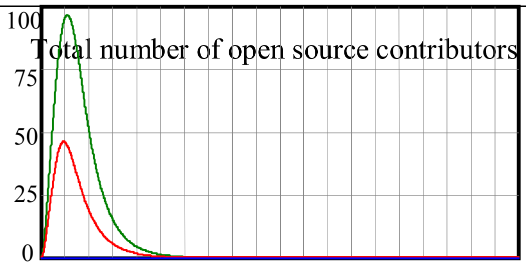


Figure 4-14: Different level of openness: Base Case- Open source contributors

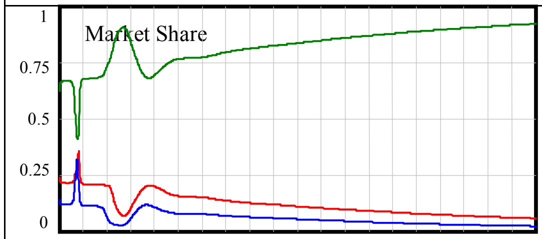


Figure 4-15: Different level of openness: Base Case- Market Share

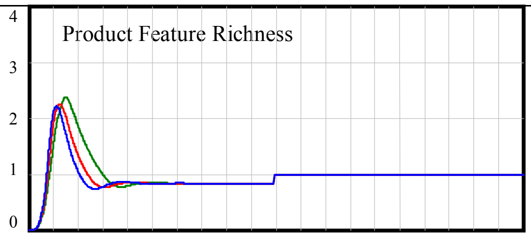


Figure 4-16: Different level of openness: Base Case- Feature richness

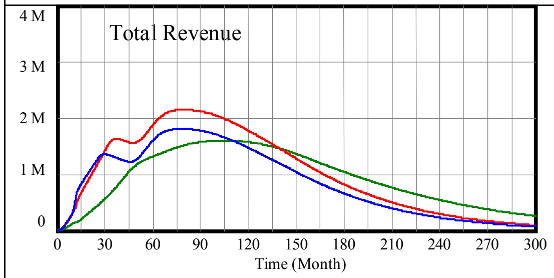


Figure 4-17: Different level of openness: Base Case- Revenue

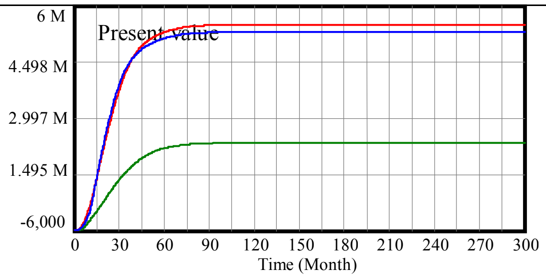


Figure 4-18: Different level of openness: Base Case- NPV

[S1]: Base Case ———
[S2]: Base Case ———
[S3]: Base Case ———

Table 4-3: Base Case behavior (S1=0, S2=0.5, S3=0.9)

4.1.2.2 Impact of aggressiveness in feature development

The aggressiveness in feature development is represented by ‘Time to close gap’ variable in the model. If any organization decides to be aggressive in developing new features and lead the market in new features they need to reduce the value of Time to close Gap variable. This way the organization is able to catch up to the markets average new features in shorter time and increase its feature richness.

In a market where are three competitors with different levels of openness (S1=0, S2=0.5, S3=0.9) being aggressive and trying to close the gap faster will impact organization S1 the most. When the organization is being aggressive in closing gap they need to hire more employees at the beginning. This would increase the employment cost of the organization. The organization with less open fraction depends mainly on their paid employees to deliver productivity. The organizations with open source developers are able to catch up using their free community resources and limit the employment cost. The excess revenue created through this aggressiveness is negated by the employment cost. The organization with higher level of openness will be impacted less in this case.

When the organization reacts less aggressively to close the gap in new features they can hire employees at a less rate from the beginning of the project and cut down the employment cost. At the same time this would decrease the revenue of the organization. As time progresses the organizations with open source method will be able to gain developers from the community to increase the development rate and catch up to the market. The revenue created through this method will be less but the profit will be higher because of the employment expense. This would positively impact the companies with less open source fraction the most because of the higher paid number of employees.

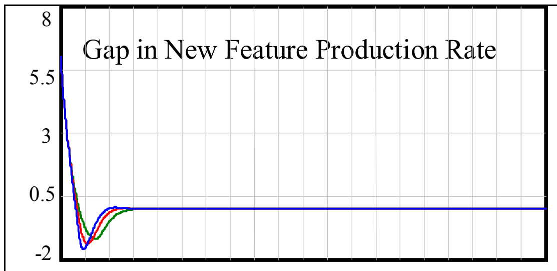


Figure 4-19: Different level of openness: Aggressive development- Gap in production rate

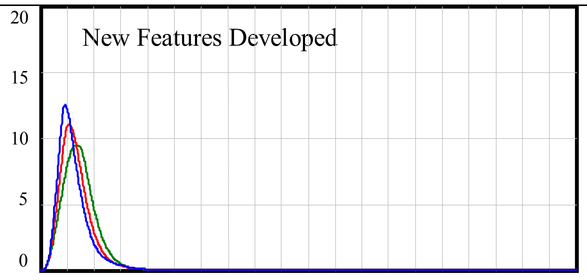


Figure 4-20: Different level of openness: Aggressive development- New features developed

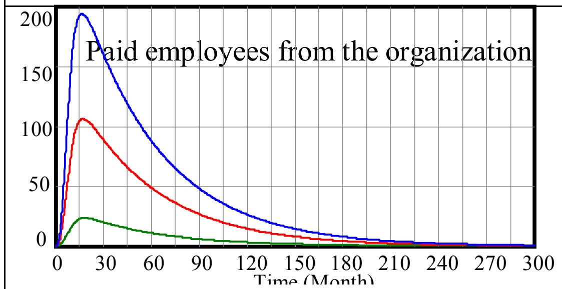


Figure 4-21: Different level of openness: Aggressive development- Paid employees

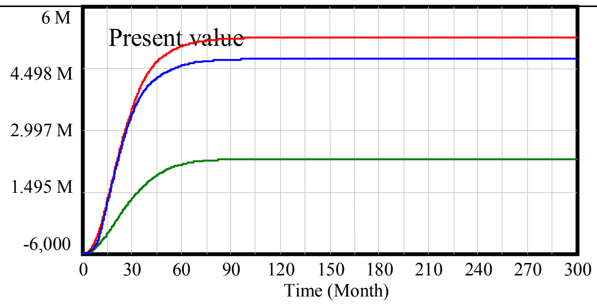
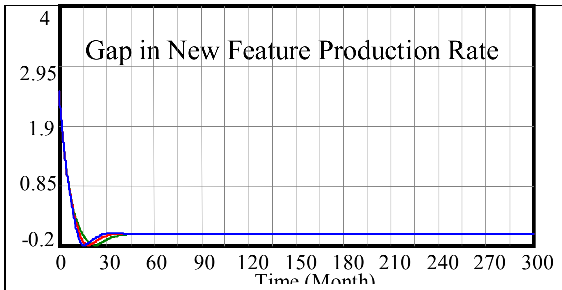
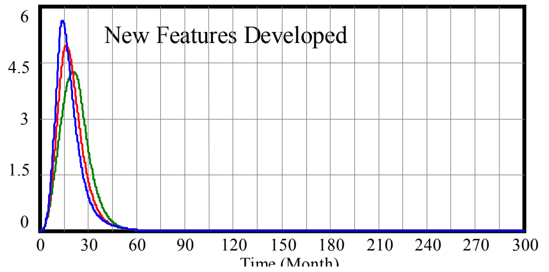
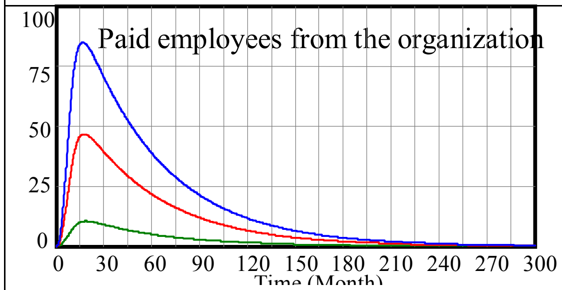
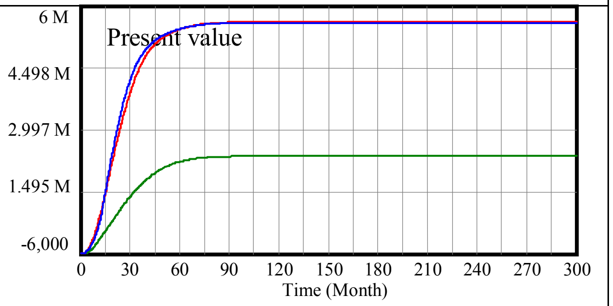


Figure 4-22: Different level of openness: Aggressive development- NPV

Present value[S1] : Less Time to close gap ———
Present value[S2] : Less Time to close gap ———
Present value[S3] : Less Time to close gap ———

Table 4-4: Aggressive Development (Time to Close Gap=5) (S1=0, S2=0.5, S3=0.9)

 **Figure 4-23: Different level of openness: Less Aggressive development- New feature Production rate**	 **Figure 4-24: Different level of openness: Less Aggressive development- New features developed**
 **Figure 4-25: Different level of openness: Less Aggressive development- Paid employees**	 **Figure 4-26: Different level of openness: Less Aggressive development- NPV**

Present value [S1]: More Time to close gap
Present value [S2]: More Time to close gap
Present value [S3]: More Time to close gap

Table 4-5: Less Aggressive Development (Time to Close Gap=12) (S1=0, S2=0.5, S3=0.9)

4.1.2.3 Impact of adoption rate of the Software

The time taken to convince a potential customer to purchase the product is important factor which needs to be considered. To understand the dynamics of the adoption rate we change the Time to adopt variable in the model.

When the market is adopting aggressively the organizations increase their sales rate from the beginning of the project. This also helps increase the revenue at the beginning of the project. As time progresses the sales rate will be less when compared to the base case since most of the market has already adopted. But the increase in revenue during the initial stage of the project will help create higher net present value for the organization (Equation 4-1). Similarly the inverse effect could be seen when the adoption rate is slower. Since the adoption rate affects the sales rate, the organizations with less openness will be affected the most.

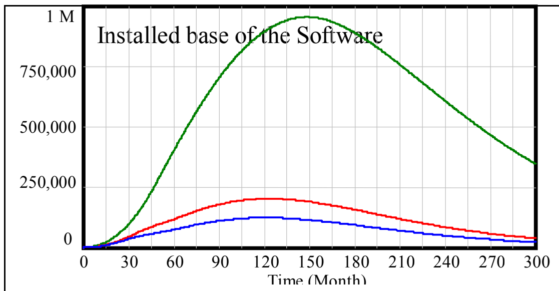


Figure 4-27: Different level of openness: Faster adoption- Installed base

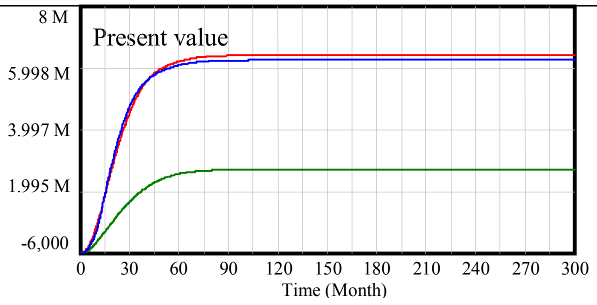


Figure 4-28: Different level of openness: Faster adoption- NPV

Present value[S1] : Faster adoption
Present value[S2] : Faster adoption
Present value[S3] : Faster adoption

Table 4-6: Faster Adoption Rate (Time to adopt=3) (S1=0, S2=0.5, S3=0.9)

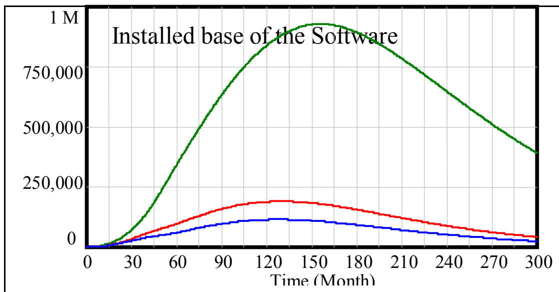


Figure 4-29: Different level of openness: Slower adoption- Installed base

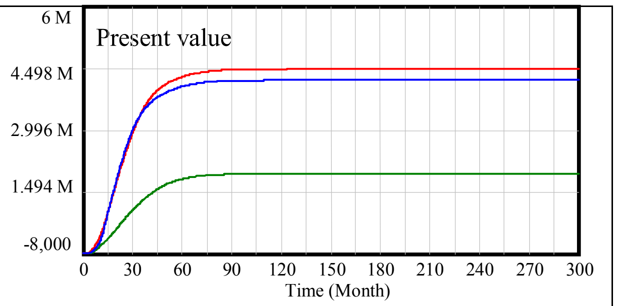


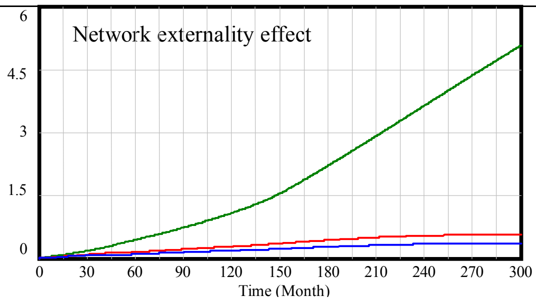
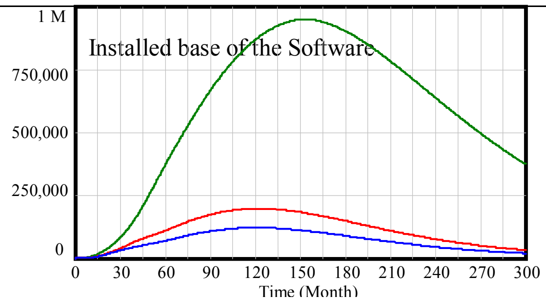
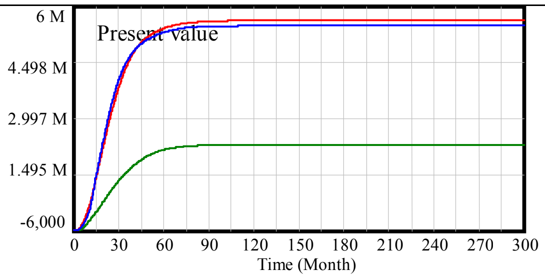
Figure 4-30: Different level of openness: Slower adoption- NPV

Present value[S1] : Slow adoption —
Present value[S2] : Slow adoption —
Present value[S3] : Slow adoption —

Table 4-7: Slower Adoption Rate (Time to adopt=8) (S1=0, S2=0.5, S3=0.9)

4.1.2.4 Strength of Customer Network

The strength of customer network depends on how many people base their purchasing decision based on previous users of the product. When the 'Sensitivity of NE' variable in the model in increased it indicates a strong customer network whereas decreasing this variable indicates that most people do not base their decision on previous users. In a sensitive market organization with larger installed base will be able gain more market share through its already existing customer base. Since having higher open source ensures larger installed base through lower price, organization S3 was able increase its revenue further by having larger customer base. In a less sensitive market organization S3 will lose revenue compared to its base case due to the lower customer base growth rate.

 **Figure 4-31: Different level of openness: Less sensitive market- Network Externality**	 **Figure 4-32: Different level of openness: Less sensitive market- Installed Base**
 **Figure 4-33: Different level of openness: Less sensitive market- NPV**	

Present value[S1] : Less Sensitive Market 🟦
Present value[S2] : Less Sensitive Market 🟥
Present value[S3] : Less Sensitive Market 🟩

Table 4-8: Less Sensitive Customer network (Sensitivity of NE = 0.4 (S1=0, S2=0.5, S3=0.9)

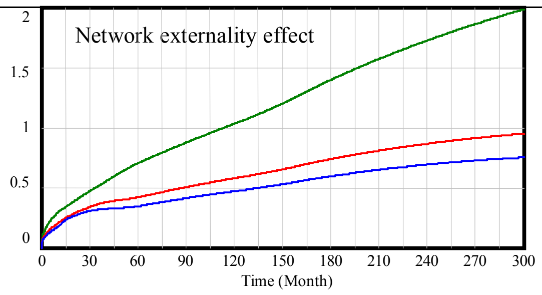


Figure 4-34: Different level of openness: More sensitive market- Network Externality

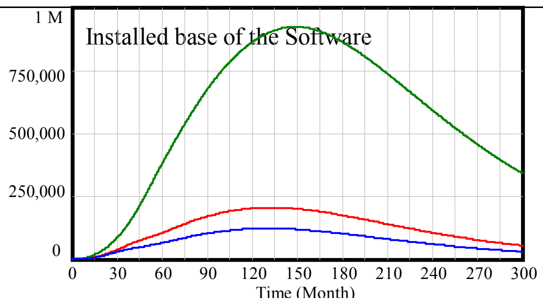


Figure 4-35: Different level of openness: More sensitive market- Installed Base

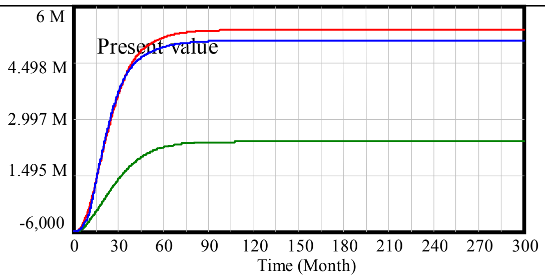


Figure 4-36: Different level of openness: More sensitive market- NPV

Present value[S1] : More Sensitive Market 🟦
Present value[S2] : More Sensitive Market 🟥
Present value[S3] : More Sensitive Market 🟩

Table 4-9: More Sensitive Customer network (Sensitivity of NE = 0.6) (S1=0, S2=0.5, S3=0.9)

4.2 Optimization

4.2.1 Constant Pricing Strategy

The software organization will decide the unit price of the software based on the level of openness. The organizations will have the freedom to reduce the price from the accepted price to increase the market share and net present value. Under the constant pricing strategy the organization has to decide on the optimal fraction of the accepted price which would produce higher net present value, at the beginning of the project and stick to it till the end. For this problem we chose a market with six competitors. The following optimization problem was formulated to find the optimal level of openness and price fraction which produce the maximum net present value of the project under the conditions captured in the model:

Optimization Problem:

Max: Net Present Value

Subject to:

$0 \leq \text{Open Source Fraction} \leq 1$

$0 \leq \text{Price Fraction} \leq 1$

Vensim uses Powell optimization method to find the maximum Net present value. This method uses line maximization (or minimization) method to find the maximum (or minimum) value of the function. The line maximization method starts at a point in a multi dimensional problem and moves in a certain direction until a local maximum (or minimum) value of the function is found. Random multiple start was used in the Vensim model to find the overall maximum value of the function. If this option is not chosen Vensim might consider the local maximum point to be the overall maximum value. The optimization was performed for 300 simulations.

Results of the optimization problem:

Maximum Net Present Value	2.139 million
Optimum Level of Openness	0.6588
Optimum Level of Price Fraction	1

Table 4-10: Base Case Optimization under constant pricing strategy

This indicates that having the price at the maximum accepted value and opening 66% of the total features to the general public would create the maximum Net Present Value for this project.

To analyze the behavior of the model with different Price fraction at the optimum level of openness the following three scenarios were chosen:

	Level of Openness	Price Fraction
Optimum Case	0.6588	1
Case 1	0.6588	0.8

Table 4-11: Change in price fraction

Decreasing the price fraction would attract more customers because of the lower price. This would increase the market share of the organization. But it would not create greater revenue because of the lower unit price. The increase in installed base in Case 1 is not sufficient enough to push the revenue over the optimum case.

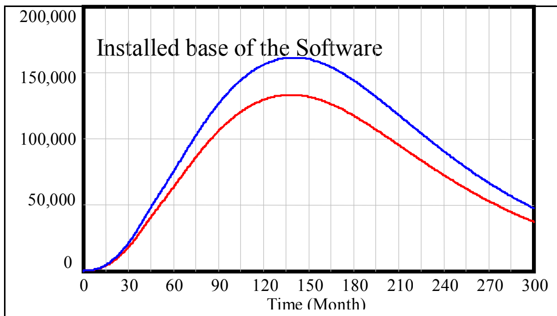


Figure 4-37: Constant Pricing strategy: Change in price fraction- Installed base

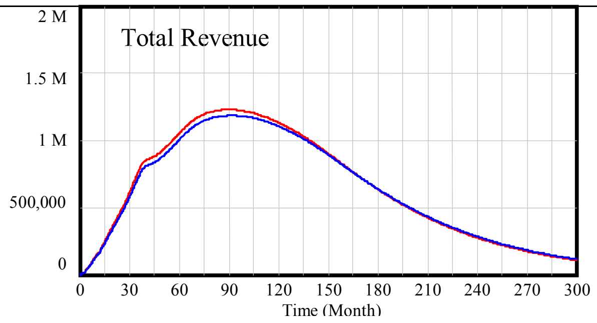


Figure 4-38: Constant Pricing strategy: Change in price fraction- Revenue

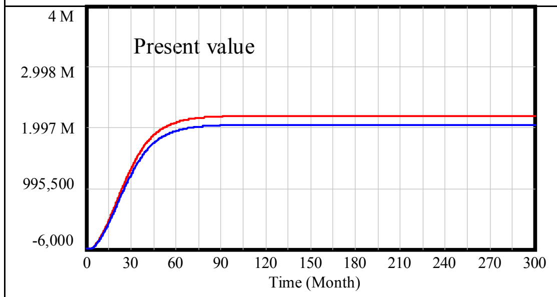


Figure 4-39: Constant Pricing strategy: Change in price fraction- NPV

Present value[S1] : Lower Price Fraction —
Present value[S1] : Optimum Case —

Table 4-12: Change in price fraction behavior

To observe the behavior of the model with the change in level of openness at the optimum price fraction the following analysis was done:

	Level of Openness	Price Fraction
Optimum Case	0.6588	1
Case 1	0.4	1
Case 2	0.8	1

Table 4-13: Change in open fraction

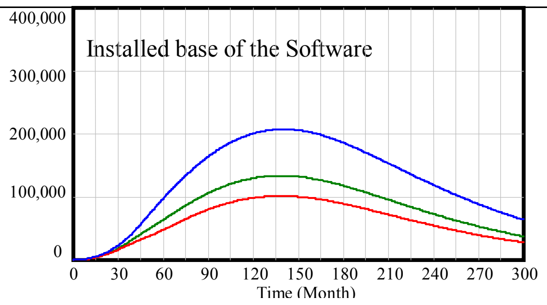


Figure 4-40: Constant Pricing strategy: Change in openness- Installed base

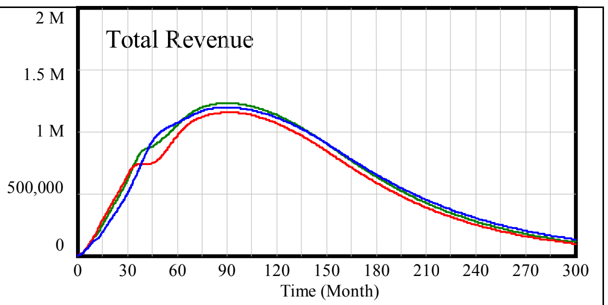


Figure 4-41: Constant Pricing strategy: Change in openness- Revenue

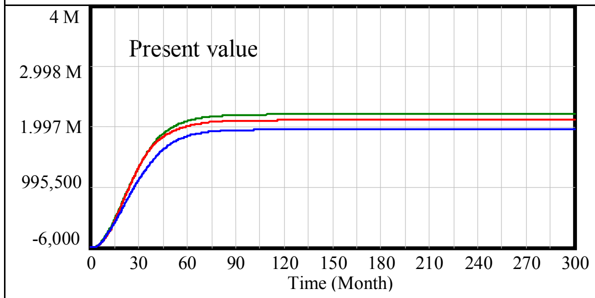


Figure 4-42: Constant Pricing strategy: Change in openness- NPV

Present value[S1] : Higher Open Fraction ———
Present value[S1] : Lower Open Fraction ———
Present value[S1] : Optimum Case ———

Table 4-14: Change in open fraction behavior

Increasing the level of openness attracts more customers because of the lower price and feature richness. The higher installed base also increases the effect of network externality and complementary product for the software. At 0.4 the network externality will be less because of the lower installed base. But at 0.4 level of openness the organization is able to make better revenue than at 0.8 due to difference in selling unit price.

4.2.1.1 Strength of Customer Network

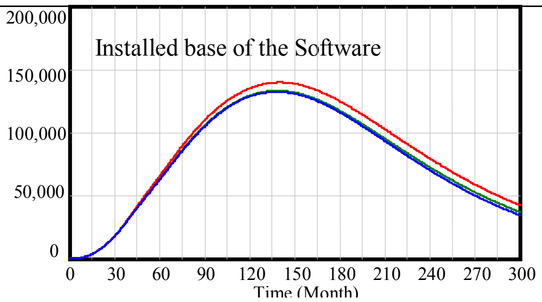
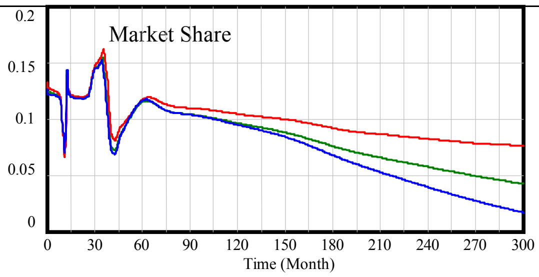
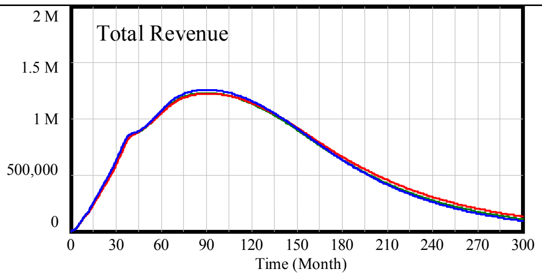
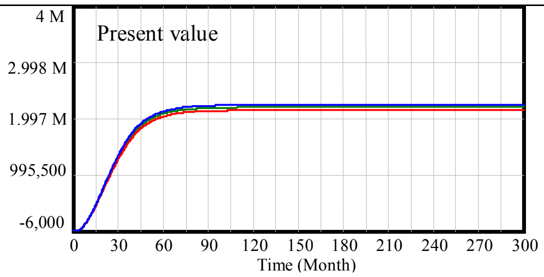
If majority of the consumers make their purchase decision based on previous customers (which indicates higher sensitiveness to network externality) it will be beneficial for organizations to increase level of openness to create larger customer base. Having a larger customer base will the organization to motivate other consumers to purchase the product.

Maximum Net Present Value	2.092 million
Optimum Level of Openness	0.6886
Optimum Level of Price Fraction	1

Table 4-15: Constant Pricing Optimization under more Sensitive Market (Sensitivity of NE =0.6)

Maximum Net Present Value	2.177 million
Optimum Level of Openness	0.6406
Optimum Level of Price Fraction	1

Table 4-16: Constant Pricing Optimization under Less Sensitive Market (Sensitivity of NE =0.1)

 **Figure 4-43: Constant Pricing strategy: Sensitivity of customer network- Installed base**	 **Figure 4-44: Constant Pricing strategy: Sensitivity of customer network- Market share**
 **Figure 4-45: Constant Pricing strategy: Sensitivity of customer network- Revenue**	 **Figure 4-46: Constant Pricing strategy: Sensitivity of customer network- NPV**

Total Revenue[S1] : More Sensitive Market
Total Revenue[S1] : Less Sensitive Market
Total Revenue[S1] : Optimum Case

Table 4-17: Constant Pricing Optimization behavior with changing strength of customer network

4.2.1.2 Impact of adoption rate of the Software

In a faster adopting market the organization will benefit by decreasing the openness. Decreasing the level of openness in such scenario will help increase its revenue through higher sales price. In a slower adopting market the organization should try to increase its openness to increase it installed base to increase its sales rate.

Maximum Net Present Value	2.589 million
Optimum Level of Openness	0.6026
Optimum Level of Price Fraction	1

Table 4-18: Constant Pricing Optimization under Faster adoption rate (Time to adopt =3)

Maximum Net Present Value	1.734 million
Optimum Level of Openness	0.7111
Optimum Level of Price Fraction	1

Table 4-19: Constant Pricing Optimization under Slower adoption rate (Time to adopt =8)

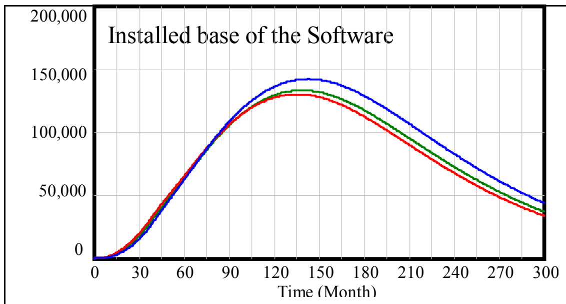


Figure 4-47: Constant Pricing strategy: Adoption rate- Installed base

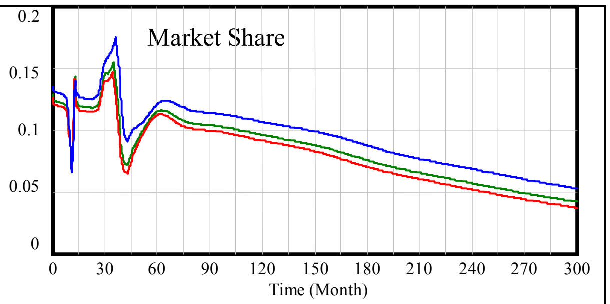


Figure 4-48: Constant Pricing strategy: Adoption rate- Market share

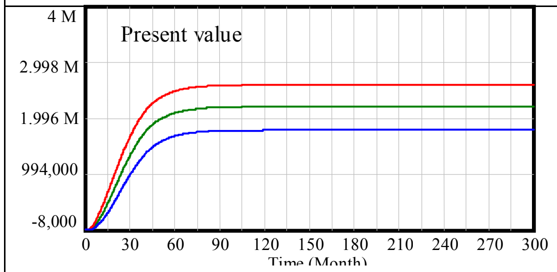


Figure 4-49: Constant Pricing strategy: Adoption rate- NPV

Present value[S1] : Slower Time to Adopt
Present value[S1] : Faster Time to Adopt
Present value[S1] : Optimum Case

Table 4-20: Constant Pricing Optimization behavior with changing adoption rate

4.2.2 Dynamic Pricing Strategy

In Section 4.2.1 optimization problem the assumption was that the unit price of the software would remain constant through out the life cycle of the project. In this section the organization decides to divide the life cycle into ten equal time frames and decide the best unit price for each time frame. The other Vensim settings are similar to the previous section.

Optimization Problem:

Max: Net Present Value

Subject to:

$0 \leq \textit{Open Source Fraction} \leq 1$

$0 \leq \textit{Price Fraction from 0- T1} \leq 1$

$0 \leq \textit{Price Fraction from T1-T2} \leq 1$

$0 \leq \textit{Price Fraction from T2-T3} \leq 1$

$0 \leq \textit{Price Fraction from T3-T4} \leq 1$

$0 \leq \textit{Price Fraction from T4-T5} \leq 1$

$0 \leq \textit{Price Fraction from T5-T6} \leq 1$

$0 \leq \textit{Price Fraction from T6-T7} \leq 1$

$0 \leq \textit{Price Fraction from T7-T8} \leq 1$

$0 \leq \textit{Price Fraction from T8-T9} \leq 1$

$0 \leq \textit{Price Fraction from T9-T10} \leq 1$

Results of the optimization problem:

Maximum Net Present Value	2.220 million
Optimum Level of Openness	0.443348
Optimum Level of Price Fraction from 0-30	0.708065
Optimum Level of Price Fraction from 30-300	1

Table 4-21: Optimization Results (Dynamic Pricing)

4.2.2.1 Strength of Customer Network

When the sensitivity of the customer network is high the companies can create larger revenue if they have a large client base. In such case they can decrease the price of their product to attract more customers. But if the sensitivity is less, increasing the price to increase revenue will be a better strategy since the sales rate created through adopted customers are going to be low.

Maximum Net Present Value	2.247 million
Optimum Level of Openness	0.443441
Optimum Level of Price Fraction from 0-30	0.745592
Optimum Level of Price Fraction from 30-300	1

Table 4-22: Dynamic Pricing Optimization under Less Sensitive Market (Sensitivity of NE =0.1)

Maximum Net Present Value	2.190 million
Optimum Level of Openness	0.443404
Optimum Level of Price Fraction from 0-30	0.649394
Optimum Level of Price Fraction from 30-300	1

Table 4-23: Dynamic Pricing Optimization under More Sensitive Market (Sensitivity of NE =0.6)

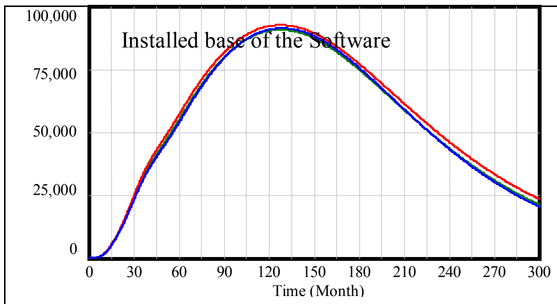


Figure 4-50: Dynamic pricing strategy: Market sensitivity- Installed base

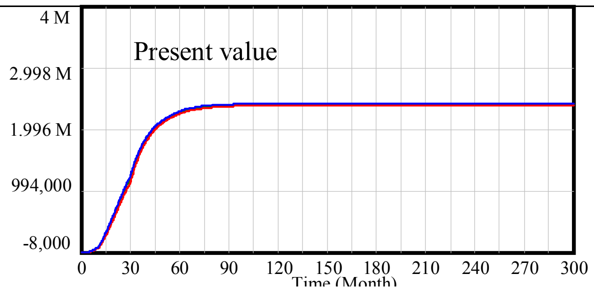


Figure 4-51: Dynamic pricing strategy: Market sensitivity- NPV

Present value[S1] : More Sensitive Market	—
Present value[S1] : Less Sensitive Market	—
Present value[S1] : Optimum Case	—

Table 4-24: Dynamic Pricing Optimization behavior with changing Market sensitiveness to NE

4.2.2.2 Impact of adoption rate of the Software

The optimal revenue in a faster adopting market will occur at higher price fraction compared to the base revenue case. Higher price fraction increases revenue due to the higher selling price. In slow adoption scenario the organization gains optimum revenue through better installed base. Higher openness and at a lower price fraction will increase the organization's installed base to create the optimum revenue in such scenario.

Maximum Net Present Value	2.731 million
Optimum Level of Openness	0
Optimum Level of Price Fraction from 0-30	0.5401
Optimum Level of Price Fraction from 30-300	1

Table 4-25: Dynamic Pricing Optimization under faster adoption rate (Time to adopt =3)

Maximum Net Present Value	1.785 million
Optimum Level of Openness	0.602489
Optimum Level of Price Fraction from 0-30	0.821018
Optimum Level of Price Fraction from 30-300	1

Table 4-26: Dynamic Pricing Optimization under slower adoption rate (Time to adopt =8)

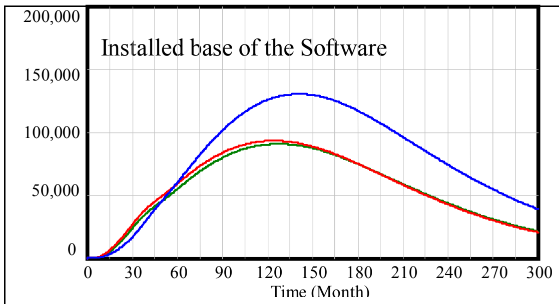
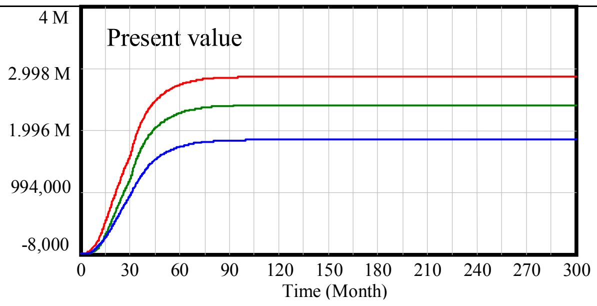
 **Figure 4-52: Dynamic pricing strategy: Adoption rate- Installed base**	 **Figure 4-53: Dynamic pricing strategy: Adoption rate- NPV**

Table 4-27: Dynamic Pricing Optimization behavior with changing adoption rate

5 Chapter 5: Conclusion

5.1 Summary of Results

Competitors with similar level of openness:

In a software market with competitors with similar level of openness, an aggressive development strategy for a particular firm will help increase the Net Present Value of that particular firm. By developing new features aggressively the firm is able to close the gap between the market's product and it's product in new features rapidly. This will increase the feature richness of the product. More potential customers will adopt this software product over its competitor's due the functionality increase.

Level of Openness of organization S1=0.2
Level of Openness of organization S2=0.2
Level of Openness of organization S3=0.2

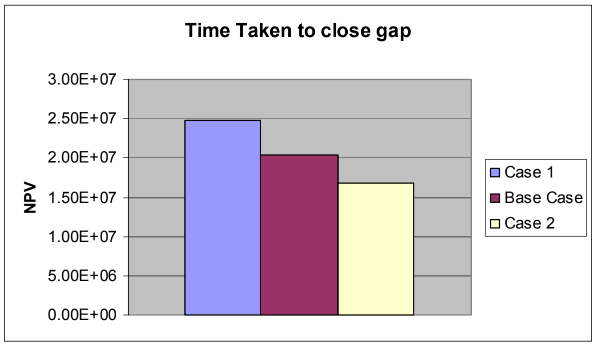
Variable	Value	Behavior (NPV)
Time Taken to close Gap	Case1=5 month Base Case=8 month Case 2=12 month	

Table 5-1: Summary of Result: Competitors with same level of openness

Competitors with different level of openness:

In a market with aggressive development such as the Antivirus market where continuous new features (software patch) need to be added, the organization with less open software will lose larger amount of Net Present Value compared to the base case. The increase in development rate will require the organizations to increase the number of developers working on the project. For a firm with less openness the majority of the employees will have to be paid by the firm. Hence the aggressive development will increase the employment cost negating the extra profit (when compared to base case) created through aggressive development.

In market with slower development rate such as the Operating Systems market the firms with less openness are able to increase their NPV more since they are able to manage their employment cost.

Level of Openness of organization S1=0
Level of Openness of organization S2=0.5
Level of Openness of organization S3=0.9

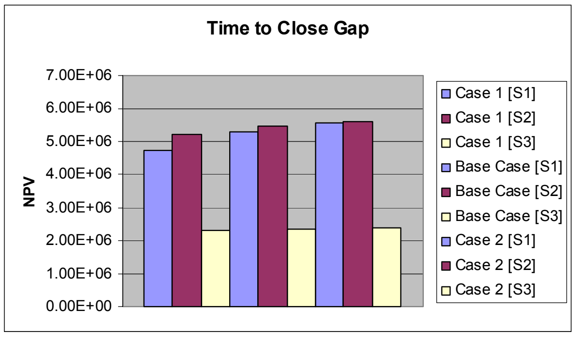
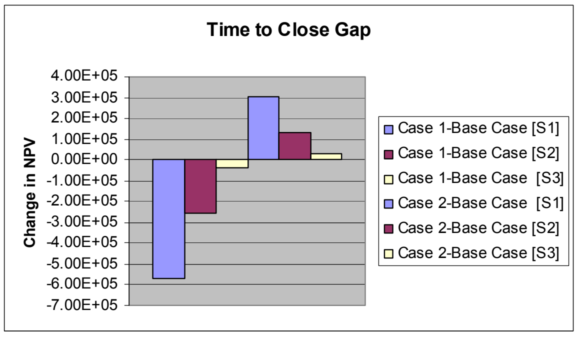
Variable	Value	Behavior (NPV)
Time Taken to close Gap	Case1=5 month Base Case=8 month Case 2=12 month	 

Table 5-2: Summary of Result: Competitors with different levels of openness: Aggressive development

The adoption rate in an anti virus market is faster than in an operating systems market. When customers buy operating systems they are concerned about the hardware configuration of the systems as well. Hence customers spend longer time in researching and deciding on operating systems compared to an antivirus. In a faster adopting market organization with less openness will be able to boost their NPV more compared to the base case due to the higher unit sales price.

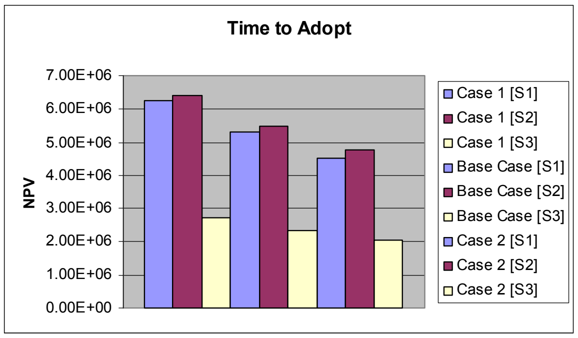
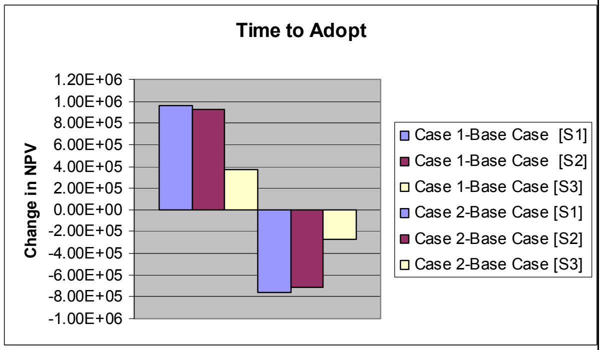
Time to Adopt	Case1=3 month Base Case=5 month Case 2=8 month	 

Table 5-3: Summary of Result: Competitors with different levels of openness: Adoption Rate

In a customer sensitive software market the potential customers base their purchase decision on what percentage of the total client-base use any particular kind of software. For example a firm looking to purchase Business Intelligence software would favor software with large client-base due to compatibility issues. In such market conditions the firm with larger open base will increase its NPV (compared to the base case) due to large market share created by lower price.

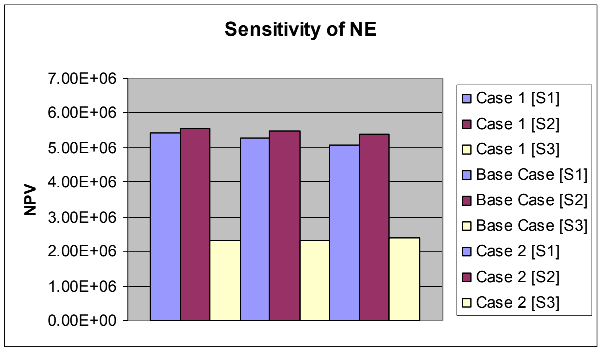
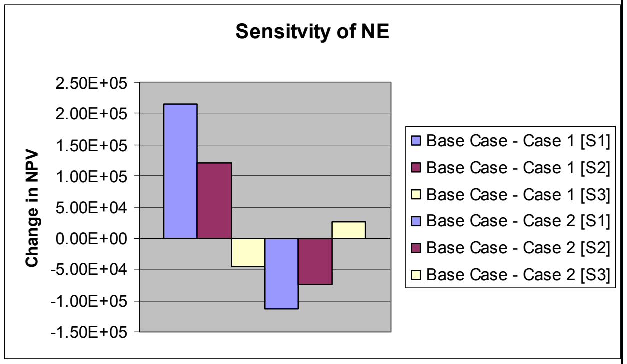
Sensitivity of NE	Case1=0.4 month Base Case=0.65 month Case 2=0.8 month	 

Table 5-4: Summary of Result: Competitors with different levels of openness: Market sensitiveness

Optimization under constant pricing strategy:

In a market with six different competitors, five of those competitor's open and pricing strategies were kept constant while firm S1's strategy was optimized under constant pricing condition. The optimal case for firm S1 was obtained by opening 66% of the total code and maintaining the maximum price.

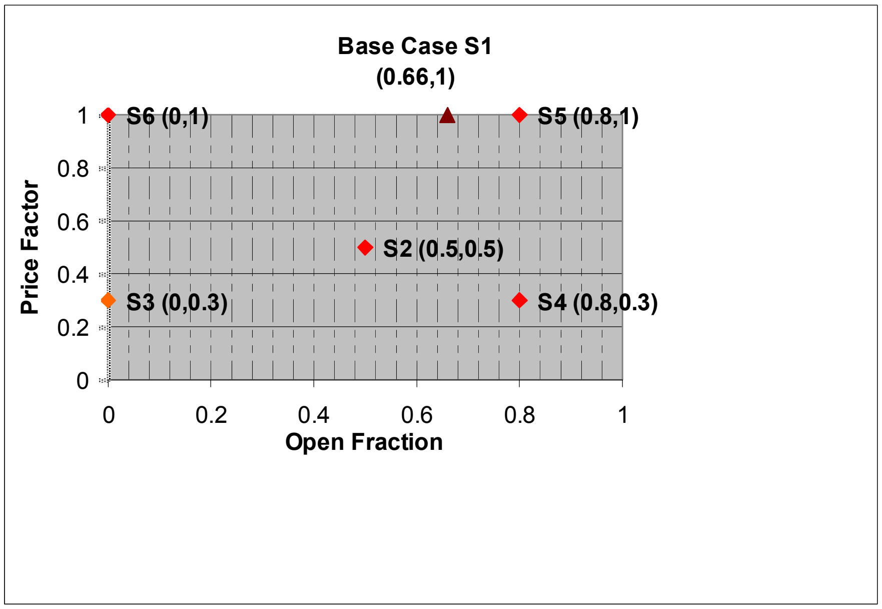


Figure 5-1: Summary of result: Constant pricing strategy- Base Case

In a less sensitive market (antivirus) under same competition, firm S1 will create larger revenue by decreasing the level of openness. Since the customer decisions are less affected by the current customer base, firm S1 can create larger NPV by increasing the sales price (this could be achieved by decreasing the level of openness). If the market is more sensitive to the current customer base (Business intelligence software), firm S1 needs to increase its client base by increasing the level of openness.

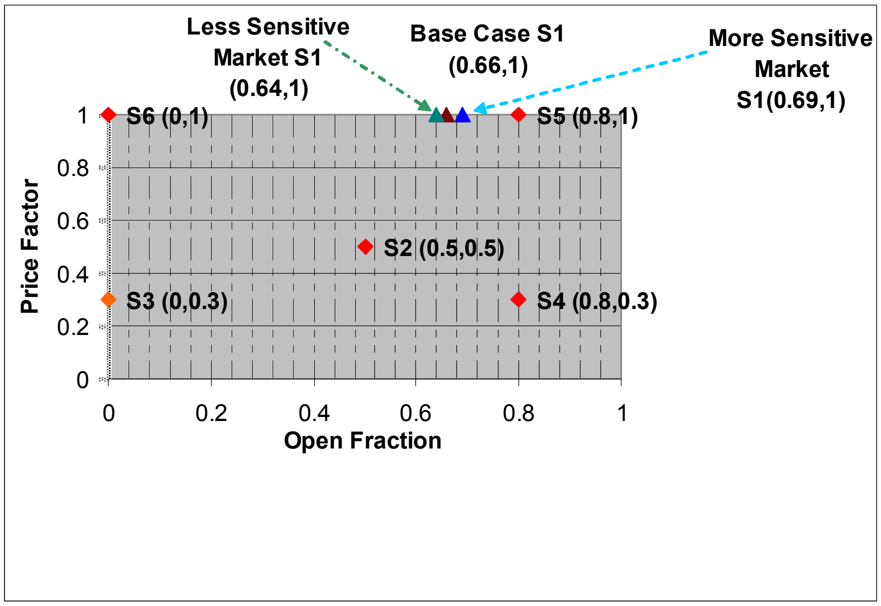


Figure 5-2: Summary of result: Constant pricing strategy- Market sensitiveness

If the customers adopt a product faster (Antivirus Market has a faster adoption rate than an Operating Systems market) the firm in that market should reduce its level of openness (compared to the base case) to increase the unit selling price to boost its NPV. In a slower adopting market software firm can increase the level of openness to increase the client base. The increase in client base will help the firm raise its market share through network externality effects.

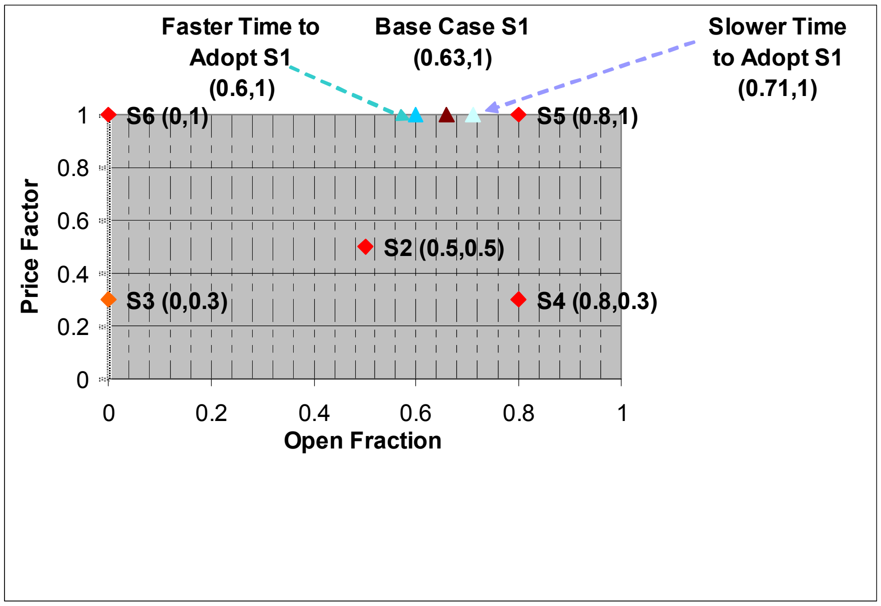


Figure 5-3: Summary of result: Constant pricing strategy- Adoption rate

Optimization under dynamic pricing strategy:

In a more sensitive market (Business intelligence software market) using the dynamic pricing strategy the firm can reduce the price (compared to the base case) of the software initially to create a larger customer base. Once the customer base is established they could increase the price to create larger NPV. If the market is less sensitive the firm can maintain slightly higher price than in the base case and after certain time increase the price to the maximum accepted price.

In a fast adopting market software firms need to be completely proprietary and maintain slightly lower price at the beginning of the project. Reducing price from the maximum accepted price will help the firm increase the popularity of the software. If the market adopts slowly the firm can increase NPV through larger customer base.

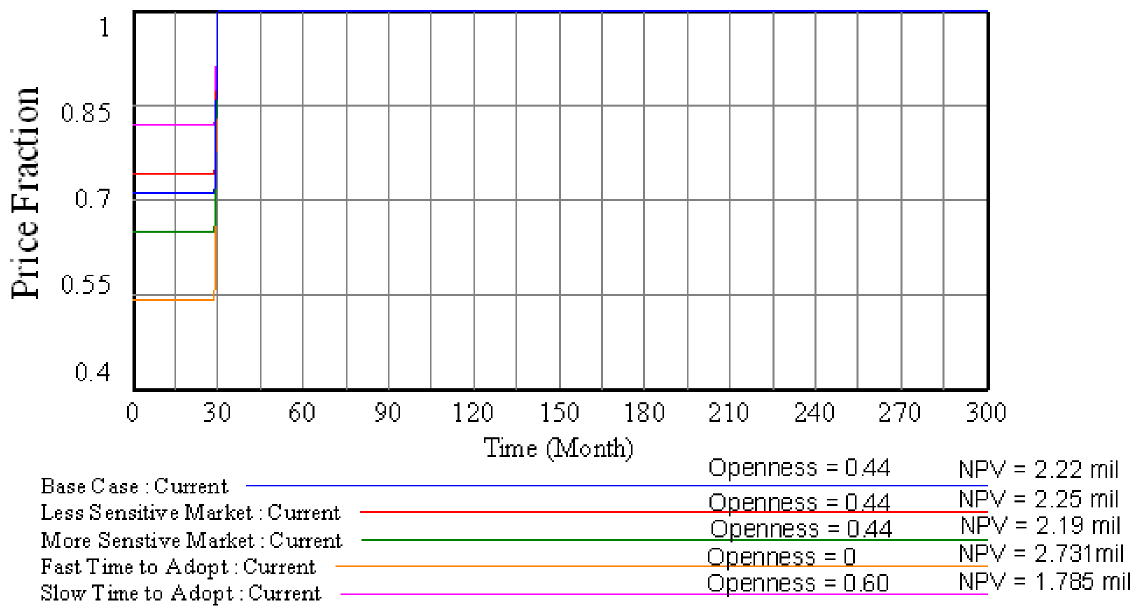


Figure 5-4: Summary of result: Dynamic pricing strategy

5.2 Policy Suggestions

The results in section 4 have provided strong argument to prove that choosing open source according to the market can help increase the revenue of the organizations. The level of openness and price should be decided based on customer adoption rate, network sensitivity, and market aggressiveness. Based on the level of openness decide by the organization they can adopt a license model to sell their product. Licenses such as BSD (Berkeley Software Distribution) and MPL (Mozilla Public License) can be used for such circumstances. BSD license permits open source codes and derived work to be incorporated with proprietary source codes. Under this license the open source code can also withheld from public when it is integrated with proprietary source codes. This license will prove to be useful for organizations willing to maintain lower level of openness. MPL is a similar license to BSD but requires all codes developed under open source method to be published at all time to the public. This license is beneficial for organizations that are willing to maintain higher levels of openness.

The software organization needs to clearly understand the sensitivity of the market to the network externality that they are competing in. For example, in a market such as the Operating System market, the consumers make decision on a systems based on how many other customers are already using that system. In such cases the market is very sensitive to the network effects. In a sensitive market it is advisable for organizations to choose a higher open source method compared to a less sensitive market. The organization can measure the sensitivity of the market for unique new products by collecting survey data from their potential customers. In a fast evolving market such as the Antivirus market the software firms can benefit form having lower level of openness and maintaining slightly higher price. Following the dynamic pricing strategy will help increase the revenue further.

5.3 Areas for Future Research

In order to make this model a business decision simulator, the marketing structure of the organization needs to be added. In the current research project this structure has not been captured. More research can be done in understanding how the organizations attract open source developers. Also the interesting and useful to learn more about the externality effect of open source developers on potential open source developers and how this impacts the dynamics of the software organization.

The current project analyzes the dynamics of the software competition with changing price fraction and level of openness. The level of openness is assumed to be decided by the organization at the beginning of the project and continues to be constant throughout the project. Introducing a varying level of openness with time, to the system will be a useful future research method. This varying level of openness will add another dimension to the optimization problem.

The software industry is not merely a product based industry. It also an industry that provides services, such as consulting, to many different industries. There are many organizations these days that provide both products and services to their clients. The revenue structure in the model could be improved by further adding structures capturing the revenue created through service provided to their clients.

Reference

Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2006). Entry strategies under competing standards: Hybrid business models in the open source software industry. MANAGEMENT SCIENCE, 52(7), 1085-1098.

Cooke, D., Urban, J., Hamilton, S., & Thompson, K. (1999). Unix and beyond: An interview with Ken Thompson. COMPUTER, 32(5), 58-+.

Cusumano, M. (2004). The Business of Software. New York: free Press.

Dedrick, J., & West, J. (2004). An exploratory study into open source platform adoption. Paper presented at the Proceedings of the 37th Annual Hawaii International Conference on System Sciences.

Dinh-Trong, T., & Bieman, J. (2005). The FreeBSD project: A replication case study of open source development. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 31(6), 481-494.

Fuggetta, A. (2003). Open source software - an evaluation. JOURNAL OF SYSTEMS AND SOFTWARE, 66(1), 77-90.

Godfrey, M. W., & Tu, Q. (2000). Evolution in open source software: a case study. Paper presented at the Proceedings International Conference on Software Maintenance. ICSM-2000.

Gurbani, V., Garvet, A., & Herbsleb. J. (2006). A case study of a corporate open source development model. Paper presented at the Proceeding of the 28th international conference on Software engineering.

Hawkins, R. E. (2004). The economics of open source software for a competitive firm: why give it away for free? Netnomics, 6(2), 103-117.

Higa, K., Sheng, O. R. L., Hu, P. J.-H., & Au, G. (1997). Organizational adoption and diffusion of technological innovation: a comparative case study on telemedicine in Hong Kong. Paper presented at the Proceedings of the Thirtieth Hawaii International Conference on System Sciences.

Kenwood, C. A. (2001). Business Case Study of Open Source Software: MITRE Corp., Bedford, MA.

Khlak, A. Agent-based Model for Economic Impact of Free Software. WILEY PERIODICALS, 8(3), 45-55.

Kuan, Jennifer W. (January 2001). Open Source Software as Consumer Integration Into Production. Available at SSRN: http://ssrn.com/abstract=259648

Lussier, S. (2004). New tricks: How open source changed the way my team works. IEEE SOFTWARE, 21(1), 68-+.

Madanmohan, T. R., & De, R. (2004). Open source reuse in commercial firms. IEEE Software, 21(6), 62-69.

Parker, G., & Van Alstyne, M. (2005). Mechanism design to promote free market and open source software innovation. Paper presented at the 38th Annual Hawaii International Conference on System Sciences.

Perkins, G. (1999). Culture clash and the road to world domination. IEEE Software, 16(1), 80-84.

Rogers, E. (1983). Diffusion of innovations. New York: Free press.

Sterman, J. D. (2000). Business Dynamics: Systems Thinking and Modeling for a Complex World. Irwin McGraw-Hill: Boston, MA.

Westarp, F. V., & Wendt, O. (2000). Diffusion follows structure - a network model of the software. Paper presented at the The 33rd Annual Hawaii International Conference on System Sciences (HICSS-33).

Wilson, G. (1999). Is the open-source community setting a bad example? IEEE SOFTWARE, 16(1), 23-25.

Appendix 1

Active Life of Development on Product=150
Units: Month
This parameter sets the development life of the product, in terms of for how long new features are added to the product in the market. Initially, the rate is as the "Base Feature Growth" but later declines, so that by the time of active life, no more new features are added to the product.

Adjustment for labor[Software]=(Desired Proprietary resources[Software]-Paid employees from the organization[Software])/Labor adjustment time
Units: Developer/Month
Adjustment in labor based on availability and need of the firm.

Adjustment for vacancies[Software]=Desired vacancies[Software] - Vacancies[Software])/Vacancy adjustment time
Units: Developer/Month
Desired vacancy rate to cover the delay in the system.

Adjustment to development rate[Software]=(Desired Development in Progress[Software]-Features Under Development[Software])/Time to adjust development rate
Units: Feature/Month
Rate of new feature development needed to cover the lag caused by the delay in the system.

Adopted Market Size= INTEG (Adoption Rate-Expiration Rate-Reinvest Rate,0)
Units: Person
Total number of customers who adopted the particular kind of software.

Adoption Rate=Potential Market Size/Time to adopt
Units: Person/Month
Rate in which people become purchase the product.

Aged features= INTEG (Features Aging Rate,0)
Units: Feature
New Feature that is now common in all software products.

Aggressive factor=1.2
Units: Dmnl
This factor represents the policy of the organization to stay ahead of the market in new features by 20%.

Aging Time=25
Units: Month

Attrition rate[Software]=Installed base of the Software[Software]/Time to reinvest
Units: Unit/Month
After a time period the software becomes obsolete. This is the rate at which consumers stop using the product.

Attrition time=12
Units: Month
Participation time duration of an open source developer.

Ave employment time=50
Units: Month
Average employment time of a developer by the firm.

Average cost of an employee[Software]=1000
Units: Dollar/Developer/Month
Average pay of an employee for a month.

Average market delay=18
Units: Month
Time realize the new feature development in the market.

Average new feature in the market=SUM(New Features Developed[Software!])/ELMCOUNT(Software)
Units: Feature
Number of new features in average software.

Average New features in the market= INTEG (Feature addition Rate-Features Aging Rate, Initial new features in the market)
Units: Feature
Total number of new features in an average software.

Average time to fill vacancies[Software]=4
Units: Month
Time taken to fill vacancy by the management.

Base conventional Market Growth=30000
Units: Person/Month
Rate in which potential customers become interested in the product.

Complementary dependent factor=0.8
Units: Dmnl
Depends on the sensitivity of the market.

Complementary goods[Software]= INTEG (Introduction rate[Software]-complementary goods aging rate[Software],0)
Units: Unit

Total number of complementary software product available in the market.

complementary goods aging rate[Software]=Complementary goods[Software]/AgingTime
Units: Unit/Month
Expiration rate.

Complementary software contribution to Utility[Software]=Complementary dependent factor*Effect of complementary goods on attractiveness[Software]
Units: Dmnl
Impact of complementary goods on the value of the software.

Conventional Feature market Release rate=New features in Market/Market aging time
Units: Feature/Month

Conventional features[Software]= INTEG (New Features released[Software],0)
Units: Feature
After certain time period the new features become common in every software.

Conversion rate of open source to developers=20
Units: Developer/Feature
Average number of open source developers who will participate in the project based on the number of open source features.

Desired Development in Progress[Software]=Desired Feature Introduction rate[Software]*Time for development[Software]
Units: Feature
Extra new features needed to cover the lag in development due to the delays in development process.

Desired Feature Introduction rate[Software]=Investment factor[Software]*Max(Gap in New Feature Production Rate[Software], 0)
Units: Feature/Month
Total number of new features accepted for development by the management based on the return ratio.

Desired hiring rate[Software]=Adjustment for labor[Software]+Employee attrition rate[Software]
Units: Developer/Month
Rate at which employees need to be hired to close the gap.

Desired Open source resources[Software]=Features Under Development[Software]*Open source fraction[Software]/Min time for Open source code development [Software]/Open Source Productivity[Software]
Units: Developer
Open source developers needed for this project.

Desired Proprietary resources[Software]=Features Under Development[Software]*(1-Open source fraction[Software])/Min Time for Proprietary code development [Software]/Proprietary Employee Productivity
Units: Developer
Number of paid developers needed for the project

Desired vacancies[Software]=Max(0, Desired hiring rate[Software]*Expected time to fill vacancies[Software])
Units: Developer
Anticipated number of vacancies.

Desired vacancy creation rate[Software]=Adjustment for vacancies[Software]+Desired hiring rate[Software]
Units: Developer/Month
Desired vacancy rate to cover the delay in the systems and the gap in employees.

Effect Life Cycle on Conventional Feature Growth=Max(0, 1-Time/Active Life of Development on Product*Tl Eff Life Cycle on Feature(Time/Active Life of Development on Product))
Units: Dmnl

Effect Life Cycle on Conventional Market Growth=Max(0, 1-Time/Active Life of Development on Product*Tl Eff Life Cycle on Feature(Time/Active Life of Development on Product))
Units: Dmnl
Software market follows a project life cycle model.

Effect of complementary goods on attractiveness[Software]=(ZIDZ(Complementary goods[Software], SUM(Complementary goods[Software!])))^Sensitivity of attracitveness to complementary goods
Units: Dmnl
Impact of complementary goods.

Employee attrition rate[Software]=Paid employees from the organization[Software]/Ave employment time
Units: Developer/Month
Rate at which employees leave the firm for other opportunities and retirement.

Employee switch=1
Units: Dmnl
If 0: constant employees If 1: choose the feedback loop employee model

Employment cost[Software]=Average cost of an employee[Software]*Paid employees from the organization

[Software]
Units: Dollar/Month
Cost occurred in a month by hiring developers.

Expected time to fill vacancies[Software]=Average time to fill vacancies[Software]
Units: Month
Time taken to fill vacancy by the management.

Expiration Rate=Adopted Market Size/Time For Market Expiration
Units: Person/Month
Rate at which the market becomes obsolete.

Expired market size= INTEG (Expiration Rate,0)
Units: Person
Customers who are not interested in the product due to expiration of the product.

Externality from users to developers[Software]=(1/(1+EXP(-ZIDZ(Indicated market Size[Software], Adopted Market Size))))
Units: Dmnl
The externality from users to developers has a S shape function. When more people use the software that product will become popular among open source developers as well.

Feasible Open source development rate[Software]=MIN(Desired Open source resources[Software], Total number of open source contributors[Software])*Open Source Productivity[Software]
Units: Feature/Month
Open source development rate will be decided based on the need of the firm and availability of the open source developers.

Feasible proprietary development rate[Software]=MIN(Desired Proprietary resources[Software], Paid Employees[Software])*Proprietary Employee Productivity
Units: Feature/Month
Proprietary development rate will be decided based on the need of the firm and availability of the paid developers.

Feature addition Rate=(Average new feature in the market/Average market delay)
Units: Feature/Month
Average new feature development rate

Feature Addition Rate based on investment return[Software]=Tl effect Return Rate on Feature addition rate(Return Ratio[Software])
Units: Dmnl
Fraction of the gap decided to close based on return ratio.

Feature Introduction rate[Software]=Feature introduction start rate[Software]
Units: Feature/Month

The number of new features decided to be added to the product in a month.

Feature introduction start rate[Software]=Max(Adjustment to development rate[Software], 0)+Desired Feature Introduction rate
[Software]
Units: Feature/Month
Features needed to be added in a month based on return ratio and development delays.

Feature richness contribution to utility[Software]=Tl effect of Feature richness on utility(Product Feature Richness[Software])
Units: Dmnl
Impact of feature richness on the value of the product.

Features Aging Rate=Average New features in the market/Market aging time
Units: Feature/Month
Average rate at which new feature become common in all software.

Features Under Development[Software]= INTEG (Feature Introduction rate[Software]-New Features development rate[Software],0)
Units: Feature
Features under consideration and development

FINAL TIME = 300
Units: Month [0,4000]
The final time for the simulation.

Gap in New Feature Production Rate[Software]=(Average New features in the market-New Features Developed[Software])*Aggressive factor/Time taken to close gap
Units: Feature/Month
Difference in new features in a month between the market and the product.

Increase in developers due to open source base[Software]=Conversion rate of open source to developers*Features Under Development[Software]*Open source fraction[Software]
Units: Developer
Number of open source developers willing to participate in the project.

Indicated demand[Software]=Market Share[Software]*Adoption Rate*Purchase per person
Units: Unit/Month
The demand forecasted by the software firm.

Indicated market Size[Software]=Installed base of the Software[Software]/Purchase per person
Units: Person
Total number of customer using the software of a particular firm.

Initial Expense=5000
Units: Dollar/Month
Basic expense of a firm.

Initial new features in the market= 30
Units: Feature

Installed base of the Software[Software]= INTEG (Sales rate of Software[Software]-Attrition rate[Software],0)
Units: Unit
The total number of software units of a particular firm being used in the market

Introduction rate[Software]=Tl Effect of Market Share on complementary goods[Software](ZIDZ(Installed base of the Software[Software], Adopted Market Size))*Normalized complementary introduction rate[Software]
Units: Unit/Month
Rate at which the complementary products are introduced to the market.

Investment factor[Software]=Investment level independent of return*Investment Weight[Software]+(1-Investment Weight[Software])*Feature Addition Rate based on investment return[Software]
Units: Dmnl
Initially the firm will want to close the gap in new feature completely even if the return ratio is less than one. But after 20 months firm will decide on the fraction of gap based on the return ratio

Investment level independent of return=1
Units: Dmnl
Fraction of gap to be closed initially.

Investment Weight[Software]=Tl Investment weight(Time)
Units: Dmnl
Investment weight with time.

Joining rate[Software]=Externality from users to developers[Software]*Increase in developers due to open source base[Software]/Time to capture popularity
Units: Developer/Month
Rate at which the open source developers join the project.

Labor adjustment time=10
Units: Month
Time taken to adjust labor.

Leaving rate[Software]=Total number of open source contributors[Software]/Attrition time
Units: Developer/Month
Rate at which open source developers leave the project.

Market aging time=6
Units: Month
Time taken for new features to become common in all software product.

Market Introduction rate=Base conventional Market Growth*Effect Life Cycle on Conventional Market Growth
Units: Person/Month
Rate in which potential customers become interested in the product.

Market Share[Software]=EXP(Utility of the software[Software])/SUM(EXP(Utility of the software[Software!]))
Units: Dmnl
The percentage of the total market covered by the firm based on the value of the software..

Maximum Price fraction[Software]=Tl accepted price based on openness(Open source fraction[Software])*Price Fraction[Software]
Units: Dmnl
The fraction of the normalized unit price decided by the management.

Min time for Open source code development[Software]=Time for development[Software]
Units: Month
Minimum time taken for the developing the open source features.

Min Time for Proprietary code development[Software]=Time for development[Software]
Units: Month
Minimum time taken for the developing the proprietary features.

NE contribution to utility[Software]=NE Dependent factor*Network externality effect[Software]
Units: Dmnl
The impact of current customer network on the utility of the software product.

NE Dependent factor=1.5
Units: Dmnl
This factor decides the impact value of customer network on the utility.

Network externality effect[Software]=(ZIDZ(Indicated market Size[Software], Adopted Market Size))^(1-Sensitivty of NE to installed base)
Units: Dmnl

The impact of the customer network depends on what fraction of the total adopted market size belongs to the particular firm. Source: Why Open source can succeed- Andrea Bonaccorsi

New Features Developed[Software]= INTEG (New Features development rate[Software]-New Features released[Software],0)
Units: Feature
Total Number of new features developed by a particular firm

New Features development rate[Software]=Feasible Open source development rate[Software]+Feasible proprietary development rate[Software]
Units: Feature/Month
The development rate depends on paid employee development rate and the open source development rate.

New Features released[Software]=New Features Developed[Software]/Market aging time
Units: Feature/Month
Rate at which new feature become common in all software.

Normal Price=600
Units: Dollar/Unit
Maximum accepted unit price of a complete proprietary software.

Normalized complementary introduction rate[Software]=0.5
Units: Unit/Month
The introduction rate of complementary goods when the firms market share is 1.

Open source fraction[Software]=0,0.5,0.9
Units: Dmnl
The percentage of the total features that are developed using open source method.

Open Source Productivity[Software]=0.01
Units: Feature/Month/Developer
Number of open source features developed in a month by one developer.

Organization hiring rate[Software]=Vacancies[Software]/Average time to fill vacancies[Software]
Units: Developer/Month
Rate at which employees join the firm.

Paid Employees[Software]=Paid employees from the organization[Software]
Units: Developer
Total number of employees working on the proprietary features of the product. They are paid by the firm.

Paid employees as constant[Software]=10

Units: Developer

Paid employees from the organization[Software]= INTEG (Organization hiring rate[Software]-Employee attrition rate[Software],0)
Units: Developer
Number of employees working on the proprietary side of the software. They are paid by the organization.

Potential Market Size= INTEG (Market Introduction rate-Adoption Rate+Reinvest Rate,0)
Units: Person
This indicates the total number of customers interested in purchasing the particular kind of software from the market.

Present value[Software]=NPV(Total Profit[Software],0.08,0,1)
Units: Dollar
Net present value of the project with 8% discount rate.

Price contribution to utility[Software]=(Maximum Price fraction[Software]^Sensitivity of price on utility)*Price Factor
Units: Dmnl
Impact of the price on the value of the software.

Price Factor=1.25
Units: Dmnl
Depends on the sensitivity of the market.

Price Fraction[Software]=1
Units: Dmnl
The organization can decide to sell the product at a lower price than the accepted maximum price. This variable gives the flexibility of changing price as a strategy to attract customers.

Price of a unit software[Software]=Normal Price*Maximum Price fraction[Software]
Units: Dollar/Unit
Selling unit price of the product.

Product Feature Richness[Software]=XIDZ(New Features Developed[Software], Average New features in the market,1)
Units: Dimensionless
The features richness compares product features to the market standard.

Proprietary Employee Productivity=0.02
Units: Feature/(Month*Developer)
Number of features developed by a paid employee in a month.

Purchase per person=1
Units: Unit/Person
Number of software units purchased by a customer.

Reinvest Rate=Adopted Market Size/Time to reinvest
Units: Person/Month
Rate in which people look to purchase the software again after being customers for certain time period.

Return Ratio[Software]=ZIDZ(Total Revenue[Software], Total Cost[Software])
Units: Dmnl
The ratio between revenue and cost. If the firm's revenue is greater than the cost return ration will be greater than 1.

Sales rate of Software[Software]=DELAY1(Indicated demand[Software], 50)
Units: Unit/Month
Consumer purchase rate.

Sensitivity of attractiveness to complementary goods=0.2
Units: Dmnl
Factor decides the strength of the complementary goods on the value of the software.

Sensitivity of price on utility=1
Units: Dmnl
Depends on the sensitivity of the market to the price of the product.

Sensitivity of NE to installed base=0.35
Units: Dmnl
This variable decides the strength of the customer base on the decision making process of a customer.

Time for development[Software]=4,4,4
Units: Month
Minimum time taken for development.

Time For Market Expiration=60
Units: Month
Time before the product becomes obsolete.

Time taken to close gap=8
Units: Month
Average time needed to close the gap in new features between the market and the product.

Time to adjust development rate=4
Units: Month

Time taken to adjust the development rate based on the delay.

Time to adopt=5
Units: Month
Time taken for people to become customers after being potential customers.

Time to capture popularity=12
Units: Month
Time taken to attract developers to work on the open source project.

Time to reinvest=50
Units: Month
Time taken for customer to make a decision on re-investing.

Tl accepted price based on openness([(0,0)-
(1,1)],(0,1),(0.140673,0.938596),(0.281346,0.833333),(0.443425,0.754386
),(0.602446,0.649123),(0.712538,0.548246),(0.785933,0.425439),(0.840979,0.307018
),(0.87156,0.175439),(0.920489,0.0789474),(1,0))
Units: Dmnl
This table function decides the fraction of the normalized maximum unit price that could be charged by the software firm based on the level of openness.

Tl Eff Life Cycle on Feature([(0,0)-
(1,2)],(0,0),(0.110092,0.114035),(0.195719,0.27193),(0.302752,0.552632
),(0.461774,0.95614),(0.574924,1.05263),(0.697248,1.02632),(0.792049,1.00877
),(1,1))
Units: Dmnl
Modifies the speed by which the product features are expanded in the market

Tl effect of Feature richness on utility([(0,0)-
(1.5,3)],(0,0),(0.119266,0.0921053),(0.307339,0.263158),(0.518349,0.539474
),(0.706422,1.15789),(0.862385,1.69737),(0.972477,2.06579),(1.08716,2.61842
),(1.20183,2.90789),(1.5,3))
Units: Dmnl
This table function determines the value of the software based on the feature richness.

Tl Effect of Market Share on complementary goods[Software]([(0,0)-
(1,1)],(0,0),(0.168196,0.0131579),(0.360856,0.0438596),(0.559633,0.109649
),(0.733945,0.219298),(0.819572,0.407895),(0.917431,0.649123),(1,1))
Units: Dmnl
The complementary software products introduce to the market depends on the market share of the organization. This table function decides what fraction of the total complementary goods will be produced based on market share.

Tl effect Return Rate on Feature addition rate([(0,0)-
(2,1)],(0,0),(0.1,0.05),(0.2,0.1),(0.3,0.2),(0.4,0.3),(0.5,0.42),(0.6

,0.54),(0.7,0.66),(0.8,0.83),(0.9,0.98),(1,1),(2,1))
Units: Dmnl
Table function decides the fraction of the gap to be closed based on the return ratio.

Tl Investment weight(
[(0,0)-(20,1)],(0,1),(0.917431,0.982456),(1.65138,0.97807),(2.69113,0.960526
),(4.15902,0.916667),(5.25994,0.881579),(6.54434,0.807018),(8.19572,0.723684
),(9.05199,0.675439),(10.0917,0.583333),(11.0703,0.491228),(11.7431,0.438596
),(12.2936,0.377193),(13.211,0.280702),(14.1284,0.188596),(14.8624,0.109649
),(15.841,0.0482456),(16.6361,0.0263158),(20,0))
Units: Dmnl
Table function decides on weight of the fraction as time progresses.

Total Cost[Software]=Employment cost[Software]+Initial Expense
Units: Dollar/Month
Total cost occurred in a month

Total number of open source contributors[Software]= INTEG (Joining rate[Software]-Leaving rate[Software],0)
Units: Developer
Total open source developers involved in the complete project.

Total Profit[Software]=Total Revenue[Software]-Total Cost[Software]
Units: Dollar/Month
Total profit in a month.

Total Revenue[Software]=Price of a unit software[Software]*Sales rate of Software[Software]
Units: Dollar/Month
Total revenue of the firm in a month.

Utility of the software[Software]=NE contribution to utility[Software]+Feature richness contribution to utility[Software]-Price contribution to utility[Software]+Complementary software contribution to Utility
[Software]
Units: Dmnl
The value of the software based on price, customer network, feature richness, and complementary products.

Vacancies[Software]= INTEG (Vacancies Creation Rate[Software]-Vacancies closure rate[Software],0)
Units: Developer
Number of vacancies in the firm.

Vacancies closure rate[Software]=Organization hiring rate[Software]
Units: Developer/Month

Rate at which the vacancies are closed.

Vacancies Creation Rate[Software]=Max(0, Desired vacancy creation rate[Software])
Units: Developer/Month
Rate at which the vacancies are created.

Vacancy adjustment time=4
Units: Month
Time to adjust vacancy.

Bonaccorsi, A., S. Giannangeli & C. Rossi (2006) Entry strategies under competing standards: Hybrid business models in the open source software industry. *MANAGEMENT SCIENCE,* 52**,** 1085-1098.

Cooke, D., J. Urban, S. Hamilton & K. Thompson (1999) Unix and beyond: An interview with Ken Thompson. *COMPUTER,* 32**,** 58-+.

Dedrick, J. & J. West. 2004. An exploratory study into open source platform adoption. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, 10 pp. Big Island, HI, USA: IEEE Comput. Soc.

Dinh-Trong, T. & J. Bieman (2005) The FreeBSD project: A replication case study of open source development. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* 31**,** 481-494.

Fuggetta, A. (2003) Open source software - an evaluation. *JOURNAL OF SYSTEMS AND SOFTWARE,* 66**,** 77-90.

Godfrey, M. W. & Q. Tu. 2000. Evolution in open source software: a case study. In *Proceedings International Conference on Software Maintenance. ICSM-2000*, 131-42.

Higa, K., O. R. L. Sheng, P. J.-H. Hu & G. Au. 1997. Organizational adoption and diffusion of technological innovation: a comparative case study on telemedicine in Hong Kong. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, 146-55.

Kenwood, C. A. 2001. Business Case Study of Open Source Software. MITRE Corp., Bedford, MA.

Lussier, S. (2004) New tricks: How open source changed the way my team works. *IEEE SOFTWARE,* 21**,** 68-+.

Madanmohan, T. R. & R. De (2004) Open source reuse in commercial firms. *IEEE Software,* 21**,** 62-9.

Parker, G. & M. Van Alstyne. 2005. Mechanism design to promote free market and open source software innovation. In *38th Annual Hawaii International Conference on System Sciences*, p 213.

Perkins, G. (1999) Culture clash and the road to world domination. *IEEE Software,* 16**,** 80-4.

Westarp, F. V. & O. Wendt. 2000. Diffusion follows structure - a network model of the software. In *The 33rd Annual Hawaii International Conference on System Siences (HICSS-33)*, p 192.

Wilson, G. (1999) Is the open-source community setting a bad example? *IEEE SOFTWARE,* 16**,** 23-25.