Copyright

by

Ju Long

2004

The Dissertation Committee for Ju Long Certifies that this is the approved version of the following dissertation:

Understanding the Creation and Adoption of Information Technology Innovations: the Case of Open Source Software Development and the Diffusion of Mobile Commerce

Committee:

Andrew B. Whinston, Supervisor

Kerem Tomak, Co-Supervisor

John Mote

Rajagopal Raghunathan

Gautam Ray

Understanding the Creation and Adoption of Information Technology Innovations: the Case of Open Source Software Development and the Diffusion of Mobile Commerce

by

Ju Long, MSW, MBA, BA

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December, 2004

Acknowledgements

I wish to thank a number of people for giving me invaluable advice and help in the years to complete the Ph.D. program and write the dissertation. First and foremost, I am sincerely grateful for my advisors Andrew B. Whinston and Kerem Tomak. Their insights and support in all these years play a critical role in my intellectual development. Dr. Whinston not only introduced me to so many wonderful research topics. He also patiently guided me through these topics and make sure I am on the right track. Without Dr. Whinston, this dissertation would be a much lesser work. Dr. Tomak has given me many insights, oftentimes from his own experiences, that help me to survive the difficult times of the Ph.D. study. His trust and support have kept me motivated. I am also grateful for the other members of my dissertation committee: Dr. John Mote, Dr. Rajagopal Raguhunathan and Dr. Gautam Ray. Their constant support and encouragement have been very important in the dissertation writing process.

I also want to thank other faculty members from the MSIS department. During these years, Dr. Prabhudev Konana, Dr. Huseyin Tanriverdi, Dr. Reuben McDaniel, Dr. Tom Shively, Dr. Eleanor Jordan, Professor Linda Bailey, Professor Elota Patton and Dr. Sharon Dunn all have generously provided their help and encouragement to me. I also benefited a lot from the interactions with my fellow students from the CREC center.

I thank my family for always being there for me. I am grateful for Juntao's love and support in all these years, which makes all these efforts worthwhile. I also thank my parents and brother for their love through good and bad times.

Understanding the Creation and Adoption of Information Technology Innovations: the Case of Open Source Software Development and the Diffusion of Mobile Commerce

Publication No.______________

Ju Long, Ph.D.

The University of Texas at Austin, 2004

Supervisors: Andrew B. Whinston, Kerem Tomak

This dissertation studies several aspects of creation and adoption of information technology (IT) innovation. In particular, my research focuses on two brand-new phenomena in IT innovation: Open Source software development model and mobile commerce. Open source is a radically new model to develop software. My dissertation explores the sustainability of open source software development model. In my research, I collect detailed empirical data of successful and less successful open source projects. I identify several important factors that may determine the success or failure of an open source project. These factors include the vital roles of core developers and the importance of publicizing a project, which have not been given adequate attention in existing literature. My work could provide a better understanding of the survival and viability of open source software development model. I also explore a more and more important business model in open source software development: enterprise open source. Unlike conventional open source development model, which depends on voluntary contributions from the developers in the community, enterprise open source is invested, developed and managed by for-profit firms. I use mathematical modeling combined with empirical case studies as the research method to study various profit models of enterprise open source. The conclusions I get are supported by the empirical data. One main implication of the research is: enterprise open source will become the main propelling force in developing open source software. It can also pose serious challenges to proprietary software development model. Having studied open source as a new way to generate IT innovation, I study how these innovations could be applied in various industries. I focus on how innovations in wireless technology can be applied in healthcare, marketing and financial services industries. I discussed in detail the available technologies and how these technologies can revolutionize the practices in the above industries. My work could be of particularly great value to business practitioners in the mobile commerce field.

Table of Contents

List of Tables xi

List of Figures xii

Introduction 1

Chapter 1 Are all open source projects created equal? Understanding the sustainability of open source software development model 4

1. Introduction 4
    1.1. Open Source: A Disruptive Paradigm 4
        1.1.1. Concept of Open Source Software 4
        1.1.2. Adoption of the OSS Paradigm 5
    1.2. Romanticized picture of Open Source software development 6
        1.2.1. The Glamorous Few 6
        1.2.2. The Lackluster Majority 6
    1.3. Research motivation 7
2. Related theories 8
    2.1. Developers from the community 8
    2.2. Core developers 10
3. Data Collection, Model Building and Empirical Analysis 11
    3.1. Data Collection Process 11
        3.1.1. SourceForge.Net as the data collection site 11
        3.1.2. Sampling Procedure 12
    3.2. Description of Variables 12
        3.2.1. Characteristics of the projects: 12
        3.2.2. Characteristics of the program development activities: ... 13
    3.3. Empirical Analysis 15
        3.3.1. Data Transformation 15
        3.3.2. Data reduction – factor analysis 16
        3.3.3. Regression analysis 19

4. Discussion of the Results and Conclusion 20

4.1. Importance of the developers' contribution from the community 20

4.2. Importance of the core developers 20

4.3. Crucial role of promoting the projects 21

4.4. Contributions of this research 21

4.5. Future research directions 22

Chapter 2 Enterprise Open Source 23

1. Introduction 23

2. The enterprise open source 25

2.1. Starting new projects 26

2.2. Sponsoring existing projects 27

2.3. Providing services and support 28

3. Enterprise OSS vendors 29

3.1. Open Source externality 29

3.2. The free rider problem 31

3.3. Model construction 32

3.4. Discussion 34

4. Enterprise Open Source software users 34

4.1. Reduce the risk 35

4.2. Model Construction 36

4.3. Discussion 38

5. Conclusions 39

Chapter 3 Exploring new innovations in mobile computing: applications of wireless technology in financial services, health case and marketing industry 41

1. Secure Financial Services: Challenges and Solutions in the New Information Age 41

1.1. Introduction 41

1.2. New Information Technologies for Future Financial Services 42

1.2.1. Wireless Information Technology 43

1.2.2. Web Services 43

1.3. Security Challenges....44
1.3.1. General Security Requirements....44
1.3.2. Special Security Issues for Wireless Financial Market....46
1.3.3. Special Security Issues for Integrated Financial Markets.47
1.4. Security Protocols....48
1.4.1. Secure Communication....48
1.4.2. Authentication and Authorization....49
1.4.3. Wireless Security....50
1.5. Conclusion....53
2. Use Wireless Web Services to Improve Patient Compliance in Clinical Drug Trials....53
2.1. Patient Compliance and Monitoring in Clinical Drug Trials....53
2.1.1. Compliance to Dosing Instructions....55
2.1.2. Compliance to symptom report diaries and surveys....57
2.2. Boost Compliance with Wireless Technologies....58
2.2.1. Wireless devices as a proactive monitoring tool....58
2.2.2. Wireless devices as a subjective data collecting tool....59
2.2.3. Pervasive user interfaces....60
2.3. Back end Services and Infrastructure....62
2.3.1. Smart back end services....62
2.3.2. Interoperable web services....63
2.3.3. Web Services Networks....64
2.4. Security....64
2.4.1. Authentication and Authorization....65
2.4.2. End-to-End security through secure XML....66
2.4.3. Wireless security....67
2.5. Conclusions....68
3. Calling All Customers....69
3.1. Innovative Data Collection....70
3.2. Potential Applications....72

3.3. Implementation Procedure...76
3.4. Challenges. 77
3.5. Endless Opportunities 77

Appendix for Chapter 1...78

Appendix for Chapter 3...85

References 89

Vita 93

List of Tables

Table 1.1: Frequency table before data transformation....................................78

Table 1.2: Frequency table after data transformation.......................................79

Table 1.3: Extraction communalities table 80

Table 1.4: ANOVA 82

Table 1.5: ANOVA 2 82

Table 1.6: Coefficient of ANOVA...83

Table 1.7: Collinearity test...84

Table 3.1: Comparison between Smart Client and WAP Client.......................86

List of Figures

Figure 1.1: Histogram of number of download before log-transformation 78

Figure 1.2: Histogram of number of download after log-transformation...........79

Figure 1.3: Scree plot...81

Figure 1.4: Latent variables and manifest variables...82

Figure 1.5: Scatter plot matrix of the component scores 82

Figure 3.1. Kerberos based financial services market. Financial services are divided into security realms. 85

Figure 3.2: Proxy servers make the WAP based wireless network and XML based web services network transparent to each other. 85

Figure 3.3: The wireless front end. Components in the dashed line box can be separate devices or combined into one single hub device................87

Figure 3.4: Mix voice and graphics user interfaces in the same multimodal wireless application. 88

Figure 3.5: The complex web services architecture at the back end. 88

Introduction

In my dissertation, I focus on several aspects of creation and adoption of information technology innovations. There have been quite a few literature in this field. However, my research focus on two brand-new phenomenon: Open Source software development model and mobile commerce.

Open source model as a radically new software development model has begun in the middle 1990. In just less than 10 years, this new model to create software has been acknowledged as one of the most influential innovation in the IT industry. Many software created by open source model have been widely adopted and used in mission critical tasks in many industries. These software are reliable, customizable, sophisticated and above all, very cost-efficient. Many industry leaders are investing a large amount of money to adopt this new model.

However, since open source phenomenon is very new, a lot of questions are not very clear to researchers. A most puzzling question is: why there are only a few open source projects succeed, while the majority of projects never do. In the first chapter of my dissertation, I examine the factors that may influence the performance of the open source projects. I collect data from 300 open source projects and identify several major factors that decide the open source project's fate. Among them, two factors are often ignored by the open source development community as well as the researchers, including the role of core developers and the importance of advocating and promoting the projects. I focus on these factors. The research could be of value to the theoretical research on open source sustainability. It could also provide useful suggestions to open source project community on how to make the project successful. Finally, to the industry investors, the factors we identified in the researchers could be used in predicting which projects are more likely to succeed and plan the investment accordingly.

This leads to our next chapter in the dissertation. Industry investors and commercial firms are another important player in the open source model. Commercial firms (enterprises) have contributed significantly to major open source software (OSS) projects through hiring OSS developers and contributing to the marketing and support efforts. However, it is still poorly understood why a for-profit firm would invest in the common good and how it avoids the free rider problem in OSS. In this paper, we explore the economic incentives and strategies for firms to invest in OSS. To our knowledge, this research is the first one to examine in detail the motivations and consequences of the enterprise open source players. We first use some of the most influential OSS projects as cases to illustrate the roles firms can play in OSS.

We then divide enterprise OSS contributors into two categories: software vendors and users. We identify that network externality and reduced future integration cost are the incentives for software vendors and users to invest in OSS respectively. Using simple economic models, we conclude that the vendors with larger market shares have more incentives to invest in the OSS. For enterprise OSS users, the decision to feedback to the community is easier reached for commodity software and fast growing experimental projects.

In the third Chapter, I focus on another new information technology innovation: mobile computing and its diffusion in various industries. This chapter is divided to three stand-alone but also interconnected parts. In part 1, I examined how wireless technology application can be applied in the financial services area and revolutionize the industry. In part 2, I studies how wireless technology can be used in health care industry and drastically improve patient compliance rate. In part 2, I propose using wireless technology in marketing research and improve the data’s validity and accuracy. These research will be of value to practitioners in multiple industry sectors. It could also contribute to the academic research in mobile commerce.

Chapter 1 Are all open source projects created equal? Understanding the sustainability of open source software development model

"It is a weirdo competitor. There is no company behind it. You don't know exactly who build it. It is free. I prefer to say: 'Look, what we have here is a small price disadvantage.' It's the first time we've had a price disadvantage." -Steve Ballmer, CEO, Microsoft, On the threat to Microsoft from Linux

1. INTRODUCTION

1.1. Open Source: A Disruptive Paradigm

1.1.1. Concept of Open Source Software

Compared with traditional proprietary software development model, the open source software (OSS) model is a radically new paradigm to develop software (Raymond, 1997; Moody, 2001; Sharma, et al. 2002). In the OSS development process, software source code is freely available for anyone to view, download, modify and re-distribute as long as it is under the same open source license (http://www.opensource.org).

Most open-source software projects rely entirely on the voluntary efforts of a community of developers to develop and the bug reports and patch reports from the end users to improve, although some projects are coordinated and led by commercial entities (Please refer to Chapter 2 in my dissertation for detailed discussion). Such a voluntary community process keeps the cost of development and testing low. The nearly zero total cost of ownership gives open source software a strong edge in the competition in the software industry. Furthermore, the concept of open source promotes the benefits of collaborative development process by ensuring that developers are able to obtain and improve the software source code, and that the software can be freely modified and expanded to meet the needs of its end users. Because of such extensive collaborations within a large scale community (including thousands of developers and end-users), open source software could achieve a higher standard of quality, compared to closed source proprietary software, and helps to ensure the long-term viability of both data and applications. In effect, in a recent study that compared the quality of the closed source software and open source software, after examining more than 6 million lines of code and tracking several programs over time, researchers find that the quality of open source software appears to be at least equal and sometimes better than the quality of closed source software code implementing the same functionality (Samoladas et al. 2004).

1.1.2. Adoption of the OSS Paradigm

Industry practitioners are increasingly acknowledging that open source software are as highly sophisticated, reliable and customizable as many of their proprietary counterparts. Major firms across industries are confident enough to run mission critical functions on open source software. For instance, in a November 2002 CIO survey of 375 information executives, 54 percent CIOs said that within five years, open source programs would be their dominant server platforms.

In the competition to benefit from the revolutionary OSS Paradigm, and also to profit from the open source movement, industry leaders, such as IBM, Oracle and Sun, have already invested tremendous amount of money in open source projects or rolled out their own open source initiatives. HP has already generated more than 2.5 billion dollars revenue from open source related services in the year 2003. Sun Microsystems is considering converting its entire package of middleware (known as the Java Enterprise System) to open source. BEA Systems, the biggest pure middleware company, has already turned one of its products over to the open-source community for further development. JBOSS, an open source project, recently has secured 10 million dollars venture capital to develop their software. IBM has also invested in more than 1 billion dollars to support open source project development. Even open source movement's most outspoken critic, Microsoft, has started to experiment with open source-like ideas such as "shared source".

1.2. Romanticized picture of Open Source software development

1.2.1. The Glamorous Few

A few projects initiated in open source community, such as GNU, Linux, Apache, MySQL and PHP, have achieved extraordinary success and are among the most prominent software used in the technology industry. Take Apache and Linux as examples: Apache, a powerful server side software, runs more than 60 percent of all websites in the world; while In the personal computer operating system market, International Data Corporation recently estimates that the open source program Linux has between seven to twenty-one million users worldwide, with a 200% annual growth rate. In August 2004, HP announced that all its lap top will be preinstalled with Linux. Many observers believe that Linux probably would be the only serious threat to Microsoft Windows' monopoly in the desktop operation system market.

1.2.2. The Lackluster Majority

However, a myth, often held by OS developers themselves, is that every open source project could achieve the great success just as the glamorous few OSS projects did. The reality is far different. Most OS projects never get off the ground. Many die at inception, while others survive, but with little momentum behind them. (Thomas and Hunt, 2004)

In a complete survey of source forge projects, where almost all of the open source projects are hosted, researchers find that among all the 46,356 projects, the median number of developers is 1, which means that there is no participation of any other developer in the project; the median number of CVS (concurrent version system, an important indicator of project activeness) is 0 and more than 90% CVS is less than 100 (Healy and Schussman, 2003). In another word, except for a few truly spectacular successful projects, the majority of the open source projects are lackluster, with no active developing activity at all.

However, the extant research on open source software has a tendency to focus just on the atypically glamorous few. There are several case studies on Apache, Linux Kernel and GNOME but no such similar studies that examine the failed projects. Therefore, without understanding the whole ecosystem that includes the vast majority of failed projects, extant research created a romanticized picture about open source. Some of the important issues, such as the sustainability of open source development in economics terms, are left unexplored.

1.3. Research motivation

Why some of the open source projects could achieve success while most of the open source projects cannot? What are the factors that could influence the success or failure of the open source projects? Our research is set to address those questions.

Our research is based on a very detailed sample of 300 open source projects. The strength of our empirical data is that we include both the leading and successful projects as well as those that are less successful. To our knowledge, very few research has done a comparison study on the successful projects and lack-luster projects. Our study is one of the first to examine the projects from both realms side by side.

The rest of the paper are organized as follows: in section 2, we review the existing theories on open source software development, especially those based on organizational theories, to decide potential factors that could influence the success of the project. In section 3, we discuss the empirical study and data analysis results. In section 4, we conclude our paper and examine the future research topics.

2. RELATED THEORIES

Compared with traditional software development model, open source model has very distinctive organizational structure, development process and culture. In the traditional proprietary software development, there is only one development entity – the software developers, while in open source development, there are two development groups: a small number of core developers (usually less than 15 people) and a large number of anonymous and volunteer developers from the community at large. In order to identify the factors that may influence the success or failure of an open source software project, we divide our discussion according to these two main development entities. We study the division of labor, co-ordination mechanisms, distribution of decision-making authority, organizational boundary and development process between these entities. These dimensions of organization have been widely used to analyze traditional organizations in organizational theories (March and Simon, 1958; Mintzberg, 1971; Nohria, 1995, Srinarayan, et al. 2002).

2.1. Developers from the community

Open source projects are drastically different from traditional software development projects mainly because they rely on a large number of anonymous developers from the community to make voluntary effort in developing the projects. These developers are organized into a very loosely centralized and networked community - a "Bazaarr" (Raymond, 1997, Tirole and Lerner, 2002). There is no formal development plan or schedule to follow strictly ((Mockus et al. 2000; Schmidt and Porter, 2001). The developers choose the projects they are interested in, decide by themselves how much effort they want to put into the development, and work according to their own schedule. This open organizational structure encourages new contributors to participate in the projects. By remaining open to new contributors, the project could have an unlimited supply of innovative ideas (Fielding, 1999; Raymond, 2001).

However, because of this uniquely open structure, attracting enough developers and keep their motivated in participating in the project becomes a crucial factor in deciding the fate of the project. This is especially important in the early stages of the development before the number of the project's developers could reach the critical mass.

Several studies have discussed how to motivate open source developers (Hars and Or 2002, Lerner and Tirole, 2002, Slaughter et al. 2003). A very important motivational factor is that developers value reputation highly (Perkins, 1999; Markus et al., 2000; Raymond, 2001). Open source community is based on meritocracy (Fielding, 1999;Masum, 2001; Raymond, 2001; Schmidt and Porter, 2001). Reputation and higher status in the open source community can not only be emotionally rewarding. It can also bring some other tangible rewards, such as new promotion and employment opportunities or learning new marketable skills (Lerner & Tirole,2000). Reputation is established through quality contributions on a consistent basis that can lead to recognition, and is the only basis of authority in the community. Therefore, creating an effective reputation mechanism that can keep developers motivated is vital to the project's survival and success.

In the open source software development process, there is not a central decision maker. Developers in the community make judgment on what tasks to do and how to do it (Fielding, 1999; Markus et al., 2000; Mockus et al., 2000). However, developers from the community use E-mail lists and online forums as the essential communication channels to reach a consensus. Therefore, to have and maintain an active email-list and online forums are crucial to keep the communication channels open for the developers at large.

2.2. Core developers

An important feature of open source software development projects is their self-governance (Markus et al., 2000; Cook, 2001; Raymond, 2001). Compared with the open source projects, most corporate projects have much stronger belief in central planning. Open source projects are different. They almost always start with a single person at their center. If, after the first couple of releases, they start to grow, people might volunteer to join (Thomas and Hunt, 2004).

It might appear at first sight that the unconstrained, quasi-anarchistic nature of the open source process leaves little scope for a leadership. This, however, is incorrect. As we discussed in the previous section, to attract as many developers from the community to participate in the project, and to keep them motivated and active is very critical for the project's success. The core developers are the ones who can ensure an effective leadership. First, the leadership sets a vision. If the leader is credible and/or the vision is compelling, this vision helps attract more developers to join in. Second, projects are partitioned by core developers into manageable units/modules and handled by individuals or teams. Coordination of developement is the responsibility of core developers. Third, the core developers need to constantly attract other programmers by advocating and promoting their projects in the community. Core developers are also responsible for new project releases and distribution.

Because the volunteer developers for each project could come from all over the world, with very diverse skills and motivations, the coordination mechanisms require an emphasis on decentralized and asynchronous communication (Fielding, 1999; Mockus et al., 2000; Asundi,2001). It is very critical for the core developers to provide broad oversight of the strategic direction for these volunteer developers.

More specifically, core developers involves in every stage of the development process (O'Reilly, 1999; Mockus et al., 2000; Scacchi, 2001; Schmidt and Porter, 2001).

First, core developers identify the projects that they will work on. It is discussed through the developers E-mail group and forums. Then a development agenda is created to coordinate the project development. Therefore, the core developers are crucial in the initial stage of the project development.

Second, core developers need to find volunteers whose experiences and interests fit the projects and encourage these developers to participate in the project.

Third, after the developers submit their report and solution to the projects, the core developers need to identify the best solution among many alternative solutions. This is very essential for the projects since the quality of the open source projects rely upon whether the developers can identify the best solutions from many different submissions.

Fourth, after several testing and revising, the core developers decide the final changes to make on the projects. Each change that core developers make will be managed and documented using CVS (concurrent version system).

Last, core developers are responsible for releasing the latest version of their projects. They are also the ones in charge of promoting and publicize the new file release.

3. DATA COLLECTION, MODEL BUILDING AND EMPIRICAL ANALYSIS

3.1. Data Collection Process

3.1.1. SourceForge.Net as the data collection site

Empirical data is collected from SourceForge.net website. SourceForge.Net is the world's largest open source software development project host site, with the largest repository of open source projects. SourceForge.net provides a centralized place for open source developers to control and manage open source software development projects. There are a total number of 89,103 open source projects hosted in SourceForge.net with more than 935,651 registered developers (as of 10/17/2004 data). It thus provides us the best research site to collect empirical data on various open source projects' performance and attributes.

3.1.2. Sampling Procedure

Our data sample consists of 300 open source software development projects hosted in the Sourceforge.Net. They are the first 300 active projects ranked by Sourceforge.Net. These 300 projects provide a rich data set that includes the most successful open source projects as well as many less prominent projects. However, unlike the other projects that are ranked lower, these projects are active enough to provide usable data on the project activities. If we randomly sample all the open source projects, the data would be un-usable since the majority of the open source projects do not have any substantial activity at all (as we have discussed in the introduction section). We cannot obtain values for important variables such as number of download and number of bug report. By focusing our sample on the first 300 projects, we could ensure our sample's validity as well as its usability.

3.2. Description of Variables

SourceForge provides very detailed information on each project's activities across the project's entire time span. We include most of the major attributes of the projects as variables in our data analysis. These variables include:

3.2.1. Characteristics of the projects:

These variables reflect the characteristics of the projects.

Development status: by Sourceforge's criteria, project's development status is divided into 7 stages: planning, pre-alpha, alpha, beta, production/stable, mature and inactive. We excluded those projects in the inactive stage.

Project life span: how long has the project been created.

3.2.2. Characteristics of the program development activities:

Our sample also collects detailed data on project's development activities. These variables include:

Number of developers: it shows how many core developers are developing this project. These core developers are different from those anonymous developers who download the program, and contribute to the project development by submitting bugs and patches. As we discussed in the previous section, these core developers are the developer(s) who develop most of the program and also promote the project in the community.

Number of messages in the forums: core developers could set up and facilitate public forums for all the developers from the community to discuss project development. It is a very important venue for the developers in the community to contribute and participate in the project development. Many development tasks are assigned through the forum discussion. Many development decisions are also decided through the forum discussion process. We use the number of messages posted in the forum as an indicator of how active the forum is. It is also used to calibrate how active the developers from the community are.

Number of mailing list: core developers could also set up multiple mailing lists to communicate with each other. It is an indicator of how active the core developers are.

Number of downloads: Number of downloads is one of the most essential variables to show how successful the project has been. Generally, the more number of downloads means a more successful project. In our study, number of downloads is used as the dependent variable in the model to measure the performance of the projects.

Number of bug report: number of bug report indicates the contributions from the developers in the open source community. A large number of bug reports generally show that the project attracts a lot of attention from the developer community. The developers make significant contributions to improve the project.

Number of patch report: similar to the variable "number of bug reports", number of patches reported by the developers in the community is also a very important indicator on how active the community developers are in the project. We assume that both variables play a significant role in influencing the project's performance.

Number of CVS report: concurrent version system (CVS) is an important tool in coordinating the core developers' development efforts. By using CVS system, the changes made by one core developer in the project will not be overwritten by the changes from another. Every core developers would be able to track the latest version of the project. The total number of CVS updates accurately indicates how active the core developers are.

Number of file releases: this variable shows how many times the core developers update the projects and release a newer version of the project. The more frequent updates show more effort from the core developers to promote the project in the community. Frequently updated projects often attract more developers to participate in the projects. This is simply because developers see the project is very active so they know their efforts could be valued. New version of files also means new challenges and tasks for the developers to work on.

Number of news release: Sourceforge.net website allows each project's core developers to post news and announcements on the Sourceforge's main page. Like releasing new files, broadcasting news is also an important way for the core developers to keep the project active and attract developers from the community to participate in the project. Thus the number of news release is an essential indicator of the core developers' effort in promoting the project in the community.

3.3. Empirical Analysis

3.3.1. Data Transformation

Before we conduct empirical analyses on the data, we need to check the data quality and see if the assumptions for the statistical procedures are validate. We use the frequencies procedure to obtain the summaries of each individual variable.

As shown on the frequency table before data transformation (Table 1.1), the mean is quite different from the median for every variable. A significant discrepancy between mean and median usually suggests that the distribution of the variable value is asymmetric. This suspicion is further confirmed by the large positive skewness, which shows that the distribution has a long right tail. For example, as shown in the histogram of the distribution of the variable Number of Download (Figure 1.1), the distribution is asymmetric, with some extreme large values on the left end and a large number of data values on the lower right end. This is because the data set includes some of the most popular projects. These popular projects have extremely large number of downloads. Therefore, the data that indicates these project activities are skewed toward the left hand side. Because of the same reason, variables, including number of messages posted on the forum, number of bug report, number of patch report, number of CVS, number of file release and number of news release, all have the similar skewness in the data distribution.

The large positive skewness could inflate the standard deviation to a point where it is no longer useful as a measure of the spread of data values. In order to increase the reliability of the data analysis, we conduct a transformation so that we can bring the distribution of the variable values closer to normal.

The log transformation is a sensible choice because the variables take only positive values and are right skewed. Table 1.2 shows the frequency table of the variables after the log transformation. From the table, we can see that the transformation has brought the distribution closer to normal: the skewness has greatly reduced, so has the discrepancies between the mean and median value of the variables. We can also see how log transformation corrected the skewness from the example of Number of Download histogram (Figure 1.2)

3.3.2. Data reduction – factor analysis

In our hypotheses development, we set to predict the performance of the projects based on a set of predictors. However, many of these variables are correlated. Therefore, after the data transformation, we use factor analysis to conduct data reduction. Factor analysis is the primarily method used for data reduction. We use factor analysis to remove redundant and highly correlated variables from the data file, and replace the data file with a smaller number of uncorrelated variables. During the factor analysis process, we will also be able to examine the latent variables that are underlying the relationships between the manifest variables.

To perform factor analysis, the specific method we employ is factor analysis with principal components extraction. The principal components method of extraction begins by finding a linear combination of variables (component) that accounts for as much variation in the original variables as possible. It then finds another component that accounts for as much of the remaining variation as possible and is uncorrelated with the previous component. By doing this, we will get a few components that could account for most of the variation.

The variables we put into factor analysis include the following ones: development status, number of developers, project life-span, number of messages posted on the forum (log transformed), number of bug report (log transformed), number of patch report (log transformed), number of CVS (log transformed), number of file release (log transformed), number of news release (log transformed) and number of mailing lists,

Extraction communalities table (Table 1.3) demonstrates the estimates of the variance in each variable accounted for by the components. The Eigen-value shows the amount of variance in the original variables accounted for by each component. The percentage of variance shows how each component accounts for the total variance in all of the variables.

Judging from the table, we extract five components (four of them with eigen-value larger than 1, and one with eigen-value as .933) and they account for a total of 10 variables. These five components explain nearly 78% of the variability in the original 10 variables. By extracting these five components, we considerably reduce the complexity of the data set by using these five components, with about 23% loss of information. After extraction, the variation is now spread more evenly over the components.

We also use scree plot (Figure 1.3) to decide the optimal number of components. The eigenvalues of each component is plotted. As we can see from the plot, the significant decrease begins at the fifth component. After the fifth component, by adding more components to the model would not significantly increase the percentage of variances explained. Therefore, it supports our decision to extract 5 components in total.

After extracting the components, the rotated component matrix helps us to determine what the components represent.

The first component is most highly correlated with a number of forum and number of patch report, and also number of bug report. All these three components are associated with the activities from the developers in the open source community.

The second component is associated with number of developers, number of CVS update, as well as number of mailing list. These three variables show the strength of the core developers of the projects.

The third component is correlated with the number of file release and number of news release. These two variables mainly describe the core developers’ activities in promoting and publicizing the project.

The fourth component is associated with one variable: development status. The fifth component is also associated with one variable only: project life span.

The relationship between these extracted components and their variables are demonstrated in the Figure 1.4. Note that in the square are the manifest variables and the latent (extracted components) are in the circle.

For each component, we also compute the component score. It is calculated by multiplying the original variable values by the component’s score coefficients. We then use the resulting five component score variables in places of the ten original variables. Using the saved component score variables is better than using the extracted components directly because they are not linearly correlated with each other, thus to avoid the linearity in the regression analysis.

However, we still look at plots of the component scores to check for outliers and non-linear associations between the components. Judging from the scatter plot matrix of the component scores (Figure 1.5), we did not see abnormalities in the component scores.

3.3.3. Regression analysis

Linear regression is used to model the value of the dependent variable – success of the projects - based on its linear relationship to one or more predictors. We assume that there is a linear relationship between the dependent variable and each predictor.

As we specified before, we use number of download as the indicator of how successful the project is. Thus, log-transformed number of downloads is the dependent variable in the model. The distribution of log-transformed downloads is closer to normal than number of downloads. And the linear regression model works better with normal variables. The independent variables, i.e. the predictors are the factors we extracted in the factor analysis. We will use the component score of each factor in the model.

The ANOVA tables (Table 1.4 and Table 1.5) reports a significant F statistic, indicating that a strong prediction power of the predictors. As a whole, the regression does a good job of modeling performance of the project. Nearly half of the variation (R Square=.465) in download times is explained by the model.

To determine whether the predictors are significant ones, we can tell from the coefficient table (Table 1.6). As we proposed, the contribution from the community, the core developers' activities as well as the project promotion all play a significant role in generating more download and make the project a success. Project's lifespan is less significant. The significant value is .169. Project's development status is quite significant. This is easy to explain since the more advanced projects usually could be able to attract more download.

Checking the multicollinearity test (see Table 1.7), we can see that for all predictors, the values of the partial and part correlation does not drop sharply from the zero-order correlation, which means that there is not significant multicollinearity.

Checking the tolerance column, the 100% of the variance means that no other predictors can explain the given predictor's variance. Therefore, we don't have multicollinearity.

4. Discussion of the Results and Conclusion

4.1. Importance of the developers' contribution from the community

Our analysis confirmed the importance of the developers' contribution from the community in deciding the success or failure of the open source projects. This result is consistent with the unique organizational structure and culture of open source development model. Unlike proprietary software development model, the essence of open source model is to depend on thousands of developers in the community to voluntarily contribute and develop the projects. Our research not only confirms the importance of the developers from the community; we also quantitatively measure the importance.

Furthermore, we identify several manifest factors that indicate the activeness of the developers contributions, including patch report, bug report as well as forum activities.

4.2. Importance of the core developers

More important in our findings is that we stressed the importance of the core developers in the projects. In existing research, core developers' role is often ignored. It is common among researchers and practitioners to believe that anonymous developers are the essential part of open source project development, and the core developers play less important or even marginal roles. However, our analysis demonstrates the crucial role that the core developers play. It is the second most important factor in deciding whether a project could success or not.

4.3. Crucial role of promoting the projects

Another important finding of our research is examining the importance of actively promoting and publicizing open source projects by the core developers. In the open source development community, thousands of projects are competing for developers' attention and contribution. For a project to success, the core developers need to actively promoting their project in the community. It is important for the projects to send signals to the developers at large to show that the project is active and developing fast. As we identify in the research, there are two important signals that the core developers could send to the community: the frequent release of new files and frequent release of the news about the projects.

4.4. Contributions of this research

This research systematically examine why certain open source projects succeed while the majority of the projects fail. Our research identified several factors that influence the performance of the projects, including the often-ignored role of core developers and the importance of promoting the projects.

First, our research could shed light to the academic research in the open source field. Our research is one of the first to compare the successful projects with the less successful ones and to identify a web of factors that could influence the performance of the projects.

Second, for industry researchers and practitioners, our research could be useful in predicting which projects have more potential to succeed, and consequently decide which projects to invest or support.

Third, for open source developers themselves, our research could also be of great value. Developers could learn from our research that what are the factors that are important in deciding the fate of their projects. For example, many developers could learn from our study the crucial role of promoting and publicizing the projects in the community, and release the project files more often. They could also try to set up a more efficient reputation system to motivate the developers from the community to participate in the projects.

4.5. Future research directions

This study is the first stage of a series research in examining sustainability of open source software development model. The current research is based on cross-sectional data. In our future research, we plan to design a time series study: we will collection data through open source projects' entire life span. We will pay special attention to the factors that could help a budding project reach a critical mass. This research is based on secondary objective data only. In the future research, we plan to conduct focus group interview and survey on open source developers, especially the core developers.

Chapter 2 Enterprise Open Source

1. INTRODUCTION

Open source software (OSS) has become one of the key components in today's information technology infrastructure. The rise of the open source movement has significantly changed the dynamic of the industry and business models of commercial software firms (Raymond 2001). Although they are generally freely available for download and redistribution, OSS have become viable alternatives to some of the large scale commercial software that cost millions of dollars to develop (Koch 2003). In addition, OSS also have the reputation of being more secure and higher quality than their commercial counterparts (Halloran and Scherlis 2002; Reasoning Inc. 2003).

Unlike small scale OSS projects that are produced by one-man effort (Krishnamurthy 2002), enterprise scale OSS require dedicated and skilled developers as well as disciplined develop processes. In fact, studies have shown that open source developers are typically experienced professional developers who often have a well established career in this industry (FLOSS Final Report 2002; Hars and Ou 2002; Hann, Roberts and Slaughter 2002).

However, it is still unclear how open source developers are motivated and compensated for their work. The open source development model has two characteristics (The Open Source Initiative 2004) that diminish the source of immediate financial compensation for OSS developers:

First, the source code of the software must be freely available. This is a basic requirement that allows the developer and user community to contribute to open source. However it enables the anyone to build the software without paying the developer any licensing fee.

Second, anyone must be allowed to modify the source code and freely re-distribute the source and the binary executables. This requirement allows the community to take project to other directions if the developers do not take the feedbacks seriously. It also ensures that the project life span will never end, even if the original developers leave, as long as there are interests in the community. However, it also makes is fairly easy for other parties to free ride over the original developer's effort and reap any commercial benefit that is associated with the software without incurring the developing costs.

Without immediate financial compensation, why do professional developers work on those projects? The OSS development model is only sustainable if its developers are properly compensated for the long term. Knowledge of developer motivations allows managers and policy makers to help build better OSS communities and also better leverage their products and services.

Some researchers have suggested that OSS developers might be motivated by non-monetary incentives, such as altruism, joy and learning, as well as the possibility of increased future compensation (Hars and Ou 2002; Hann, Roberts and Slaughter 2002). However, those studies have also shown that a large number of OSS developers, especially those working on large projects that have the greatest influences on the community, are paid to work on OSS as full time employees of for-profit firms (FLOSS Final Report 2002; Hars and Ou 2002). This trend is expected to continue as OSS proliferates in the enterprise world. For those developers, the motivation to contribute to OSS can be purely economic – to earn a salary. For enterprise sponsored large projects, the question of developer motivation and incentive is transformed to the firm's competitive strategy. The research questions we try to address in this paper are: How do firms make money from OSS? How do they avoid the free rider problem? What are factors that determine their involvement and contributions to OSS?

However, past research has paid little attention to firm's motivation and strategy in OSS community investment (FLOSS Final Report 2002; Lerner and Tirole 2002). In this paper, we will investigate the enterprise investment in open source development from a micro-economic point of view. In the next section, we will first use real world cases to discuss different ways enterprises can contribute to open source. Then, in the two sections that follow, we will analyze the economic incentives for enterprises to participate in the open source community. We found it is useful to distinguish firms that build business models on OSS (OSS vendors) from those that use OSS in internal IT projects (OSS users). Based on the models we propose, we will investigate the conditions and optimal amount of contributions from each firm.

2. THE ENTERPRISE OPEN SOURCE

A typical OSS project has three important development stages in its life cycle: the start of the project including the development of the first version of the code; the iterative improvement process based on feedbacks on the pre-release candidates from the user community; and finally the major release of the software. Of course, the improvement and release stages can be inter-winded since users of the major release could also contribute bug fixes and feature enhancements to the project although they are much less frequent than those from pre-release beta users.

In the following three sections, we will use some of the most influential OSS projects as cases to show that commercial firms can contribute to OSS projects in all the above development stages. In addition, we investigate and derive the characteristics of each type of contribution.

2.1. Starting new projects

Firms can release software that they have developed using their own resources into the community and start an OSS project around it. Examples of such OSS projects include the following.

The Eclipse Project. The Eclipse Foundation is one of the largest OSS development collaboration in the world. More than 50 member firms contribute OSS code to the projects hosted by it. The flagship product, the Eclipse IDE, is one of the most popular developer productivity tools and user interface toolkits in the Java community. The Eclipse Consortium is established in 2001 after a 40 million dollars worth of software donation from IBM (IBM 2001). After the donation, IBM continues to lead and support the development of Eclipse OSS by contributing developer time and other engineering resources. IBM's commercial developer offerings, the WebSphere Studio line of products, are built upon the Eclipse platform.

The OpenOffice Project. The OpenOffice Suite is a business productivity suite similar to the Microsoft Office. It is available for free on Windows, Linux, Solaris and Mac OS X platforms. It is the most popular MS Office alternatives for corporations that need to cut software licensing cost or wish to use non-Windows based desktop solutions. OpenOffice was originally developed as proprietary software. Sun Microsystem bought the original developer of OpenOffice in 1999 and donated most of the code to establish the OpenOffice.org open source community. Many of original OpenOffice developers work on the OSS project as Sun employees after the donation. Sun offers a commercial version of OpenOffice with proprietary add-ons and support contracts.

Other similar OSS projects started by commercial firms include the NetBeans tools project from Sun Microsystem, the Beehive project from BEA and the Darwin project from Apple Computer. Even Microsoft, the most vocal OSS critic, has donated several of its own small projects to the open source community (Kerner 2004).

The common characteristics of enterprise-initiated OSS projects are that they are started by a donation from a commercial software vendor, are actively maintained by the vendor's paid employees after the donation and are the basis of other commercial offerings from the vendor.

2.2. Sponsoring existing projects

Firms also involve in open source development by sponsoring existing projects. The more popular the project grows, the more likely it will attract contributions from enterprises. The Linux and Apache projects are two examples of such enterprise involvement.

The Linux Project. The Linux Operating System project was started as a one-man hobbyist project. As it becomes increasing popular in the late 90's, many firms start to contribute to its development. Today, IBM, HP, RedHat, Novell and many other firms hire core Linux developers to work on Linux full time. Those firms also contribute to marketing and promoting the Linux brand. IBM alone spent one billion dollar in promoting Linux related products and services in 2001 (Wilcox 2000). IBM and HP have both made multi-billion dollar profits on Linux products and services in 2003 (Lyman 2004).

The Apache Project. The Apache project started as academic project for HTTP web servers. Over the years, it grew into the most popular web server on the Internet. The Apache Foundation also oversees several dozen sub-projects that have high impacts on the Internet server technologies. IBM and Sun hire many core Apache developers to work on Apache open source software. In 2004, IBM donated 85 million dollar worth of proprietary code in its Java embedded database product, Cloudscape, to the Apache Foundation (Sherriff 2004).

In addition to large contributions from big firms, OSS projects thrive on contributions from smaller firms. Projects like Linux and Apache are used in daily operations in many small to middle sized businesses. IT professionals working in those smaller firms help to discover bugs, develop patches and enhance features. They work on improving the open source software in their firm paid time as part of their job.

Firms that contribute to existing OSS projects are typically those who have product offerings based on the project or those who have used the OSS internally.

2.3. Providing services and support

A successful OSS project itself can spawn for-profit businesses. OSS developers could form firms to capitalize on the popularity of the software by offering service and support to its users.

The MySQL Project. The MySQL database project produces the most popular open source relational database software. It is widely adopted by many enterprise users. The MySQL Inc. is founded by core developers of the MySQL project to provide 24/7 product support, product customization and training to its users.

The JBoss Project. The JBoss application server is the most popular Java application server software in use today. The JBoss Inc. hires core developers in the JBoss community. They offer services, documentation and training to JBoss users.

The ”free software commercial support” model is especially suitable for business software since their users are risk averse and are more likely to purchase. The income from commercial services subsides the OSS developer’s salary and other development costs.

3. Enterprise OSS vendors

Although it is not practical to charge a license fee for OSS, commercial software vendors have developed several business models to make money indirectly from OSS. Those business models are based on the externalities of OSS and they apply to all three types of enterprise OSS contribution discussed in the last section. In this section, we will discuss what are those externalities and how they provide incentives for enterprises to invest in the public good.

3.1. Open Source externality

Based on their survey results, FLOSS Final Report (2002) indicates that firms can profit from OSS as a distributor, a retailer, an enabler or a service provider. All those business models are based on the externality effect of OSS.

OSS is typically part of a complete technology solution. For any particular piece of OSS, there are additional hardware, software or service packages that can work with it and make it more useful. That is particularly the case if the OSS is developed by volunteers, who are not bound by business contracts to fix bugs or provide support after the software is released. The externality effect between OSS and complimentary services has been exploited by many technology vendors.

A firm could invest in an OSS project and use the freely available software to increase their market share in a particular business sector. Then, it can sell the complementary products to recoup the cost it invested in the OSS project. Based on previous studies (FLOSS Final Report 2002; Raymond 2001) and our own observations, we summarize important OSS externality effects in the following list.

Pre-configured hardware and software bundles: Dell, HP and IBM all sell computers pre-installed and pre-configured with the Linux operation system. The firms went through the effort to choose and test the compatible hardware components and offers guarantees on their products.

Packaged software solutions: Linux distribution vendors such as RedHat, Novell and Sun sell software bundles that include the free Linux operation system and other OSS or proprietary software developed by themselves or by third parties. In the bundling process, their employees fix the compatibility bugs between the OSS components and add important features demanded by their customers. The customers buy the bundle knowing that it is already tuned and tested for its specific market.

Professional services and customization: JBoss and MySQL hire the core developer of their flagship OSS to provide premier level services to paid customers. Those services include: training and technical support by developers with direct knowledge of the software; and priority handling of bug fixes and feature requests etc.

Add-ons to OSS platforms: As we discussed in the previous section, IBM started the Eclipse Consortium via a large donation of software to the OSS community. Many IBM WebSphere software products since then are based on the Eclipse platform with IBM proprietary add-ons. The same types of proprietary add-ons have been developed by Sun for the NetBeans platform etc.

Basic research: Firms can use OSS projects to implement the proof-of-concept systems for research projects. As long as the firm retains the intellectual property of the research results (e.g., patent protection), the OSS project receives rigorous peer review to improve the research and promotes the related commercial products at the same time. The IBM alphaWorks hosts many small research OSS projects from IBM scientists and engineers.

3.2. The free rider problem

Although the firms that leverage the externality of OSS have strong incentives to invest in the community, the ”free rider” problem has to be addressed first. Since OSS is available for anyone to see and study, it is not difficult for a free rider that has not invested in the development to come up with the same complementary products and sell them for less. In fact, all the above externality based business models could potentially be plagued by free riders. The following are some representative free rider scenarios from real world OSS projects.

The mere fact that there are numerous Linux-based hardware sellers and numerous Linux distribution providers suggests that some firms might contribute less to the community and free rides on other firms’ effort in improving the Linux platform.

Both JBoss and MySQL have encountered third party developers who no longer contribute to the OSS project to set up consulting services that competes with them.

There are numerous small firms specializing in providing Eclipse plugins for various development tasks. Many of such products directly compete with IBM’s WebSphere line of products.

In each of the above cases, the OSS contributing firms have competitive edges over non-contributing firms since they have more technical know-hows and better known brands.

Although free riding is possible in the OSS world, it might not be the best strategy for the firm to maximize its profits. One of the strongest incentives for firms to invest in OSS is to increase the user base and hence create a ”bigger pie” for the externality market. It might not be a good idea to let your competitors decide the size of your market. To investigate this question fully, we need an analytical model to calculate the optimal strategy for each firm.

3.3. Model construction

Without losing generosity, we assume that there are n firms in the externality market of a particular Open Source project. Every firm sells the exact same complementary product (e.g., service or add-on) at a unit price p. The market share of firm i is denoted as S_i. In this analysis, we take a snapshot of the current market and ignore the potentially complex relation between S_i, C_i and the nature of the firm. The Open Source software is being cooperatively designed with firm i contributing C_i. We assume that total size of the externality market is a function of the total investment in the Open Source project $f(\sum_{j=1}^{n} C_j)$. Function f monotonically increases. For each firm i, the optimal level of investment in the Open Source project C_i^* is determined by maximizing the following function:

$$\max_{C_j^*} (p \times S_i \times f(\sum_{j=1}^{n} C_j) - C_j$$

The first order condition is:

$$p \times S_i \times \frac{\partial f(\sum_{j=1}^{n} C_j)}{\partial C_i} - 1 = 0$$

Proposition 1: If f is concave, the optimal Open Source investment of each firm increases with its market share S_i and the price of the complementary product p.

Proof: Since C_i is the only interesting variable that we are concerned here, we have

$$\frac{\partial f(\sum_{j=1}^{n} C_j)}{\partial C_i} = \frac{\partial f(\sum_{j=1}^{n} C_j)}{\partial \sum_{j=1}^{n} C_j}$$

Since f is concave, we have

$$\frac{\partial^2 (p \times S_i \times f(\sum_{j=1}^{n} C_j) - C_i}{\partial^2 C_i} = p \times S_i \times \frac{\partial^2 f(\sum_{j=1}^{n} C_j)}{\partial^2 \sum_{j=1}^{n} C_j} > 0$$

Hence, the first order condition corresponds to the maximum value of the profit function. For simplicity, let's assume that

$$g(\sum_{j=1}^{n} C_j) = \frac{\partial f(\sum_{j=1}^{n} C_j)}{\partial \sum_{j=1}^{n} C_j}$$

Solving for the first order condition,

$$g(\sum_{j=1}^{n} C_j) = \frac{1}{p \times S_i}$$

And hence,

$$C_i^* = g^{-1}(\frac{1}{p \times S_i}) - \sum_{j \neq 1} C_j$$

Since the function f is concave, both g and g^{-1} are strict decreasing. That indicates that C_i^* increases with S_i and p.

Proposition 2: If f is convex, there is no optimal level of Open Source investment for any firm.

Proof: Since f is convex, we have

$$\frac{\partial^2 (p \times S_i \times f(\sum_{j=1}^{n} C_j) - C_i}{\partial^2 C_i} = p \times S_i \times \frac{\partial^2 f(\sum_{j=1}^{n} C_j)}{\partial^2 \sum_{j=1}^{n} C_j} < 0$$

Hence, the first order condition corresponds to the minimum value of the profit function. We cannot maximize the profit function through Open Source investment under this condition.

Proposition 3: If the function f takes the form $f(x) \propto x^{\alpha}$, the firm i only invests in Open Source when $(\alpha - 1) \times p \times S_i > 1$.

Proof: In order for a firm to invest in Open Source, its profit must grow as a result of increasing investment C_i. Hence, we must have

$$\frac{\partial p \times S_i \times (\sum_{j=1}^{n} C_j)^{\alpha} - C_i}{\partial C_i} > 0$$

That directly results in $(\alpha - 1) \times p \times S_i > 1$.

Lemma 3.1: If $\alpha \le 1$, no firm will invest in the Open Source project and all will be free riders. This is the case when the Open Source project market size actually shrinks with additional investments.

Lemma 3.2: For $\alpha > 1$, only firms with large market shares will have incentives to invest in the Open Source project.

Proof: For $\alpha > 1$, the condition in Proposition 3 directly leads to

$$S_i > \frac{1}{(\alpha - 1) \times p}$$

3.4. Discussion

The firm's strategy in investing in OSS depends greatly on the how the externality market size responds to investments in the underlying platform. At the beginning of a popular project, the adoption grows following the power-law with since the larger the installed base, the wider the word can spread. The externality market size has to respond to new investments at a powerlaw index $\alpha > 1$ to be sustainable. According to proposition 3, in this stage, the firms with large market share are more likely to benefit from investment in Open Source and smaller firms are typically free riders.

As the product matures, it will inevitably reach the concave part of the growth curve since the new investment will have less and less effect on expanding the market. At this stage, all firms have incentives to invest in the Open Source project and their optimal level of investment increases with their market share.

4. Enterprise Open Source software users

FLOSS Final Report (2002) suggested that there are four main motivations for firms to become OSS users: standardization of software platform; low-cost component; strategic consideration and enabling compatibility. Archiving those goals depend on the corporative efforts of the OSS community. Adopting the software is only first step in building a healthy OSS community. In fact, the OSS development model is most successful with frequent feedbacks and contributions from a large user community (Raymond 2001). In the world of OSS, the line between the user and the developer is blurred. OSS users can feedback to the development process in several ways:

They can participate in the community mailing lists and forums to make suggestions and help new users.

They can help fix bugs and send in patches to improve the quality of the software.

They can contribute to new features and other enhancements of the software.

Some users can even become part of the core developer team and contribute significantly to the software development. If a firm uses OSS internally, it is a user member in the OSS community. The primary incentive for it to make bug fixes, security patches and feature enhancements (”patches” for the rest of the article) developed by its internal staff available for free to the community at large is to reduce the maintenance cost in the future. In this section, we will establish a model to analyze the important factors that lead the firm to decide whether and when to release their patches.

4.1. Reduce the risk

For modern software development projects, the costs of maintenance and integration often far exceed the cost for writing the software itself. If a firm develops a patch for an Open Source software and keep it proprietary, the patch will need expensive ongoing service for the rest of its proprietary life. Since the developers in the community are not aware of the inner workings of the proprietary patch, they can often break the patch functions in the subsequent versions of the software. On the other hand, when a patch is released to the Open Source community, the developers in the community will be responsible of keeping it integrated with future versions of the software. The proprietary patch excludes the firm from leveraging the key strength of the Open Source development model and could represent a big cost to the firm in the long run.

However, if the patch would allow the firm to gain competitive edges over its commercial competitors, it is another story. The firm might keep the patch proprietary and use the additional profit to offset the patch maintenance cost. For example, if a Internet data archive firm developers a proprietary extension to the Open Source MySQL database to improve its performance over large datasets, it might keep the patch proprietary to avoid free riding from competitors.

4.2. Model Construction

Let's assume that the proprietary patch generates a constant cash flow of m per unit time until it is released by the firm or independently developed by other developers in the community. The main version of the software is released at a time interval of Δt and the firm has to spend $f(n)$ amount of money to re-integrate the patch into the software for the nth version released after the patch is developed. The function $f(n)$ is undefined for non-integer values. The time for the nth software release is t_n = n × Δt. We assume that function $f(n)$ monotonically increases with n since the integration cost increases with complexity of the software, which in turn, increases with every release. The total profit for the firm at the nth release time is $p(n)$ which is described in the following formula. The firm's strategy is to find the optimal time t_n to release the patch that will maximize $p(n)$.

$$p(n) = m \times n \times \Delta t - \sum_{i=0}^{n} (f(i))$$

At the initial time, we have $f(n) = 0$ and $p(n) = 0$. Based on this simple model, we can quickly reach three important conclusions.

Proposition 4: Function p is always negative and it monotonically decreases with n if m × Δt ≤ $f(1)$.

Proof: For each increase in n from n to n + 1, we have

$p(n+1) - p(n) =$ m × Δt − $f(n+1)$

Since $f(n)$ is a monotonically increasing function, we have $f(n+1) \geq f(1)$ ≥ m × Δt. That yields $p(n+1) - p(n)$ ≤ 0 and hence $p(n+1) \leq p(n) \leq p(0) = 0$. Under this condition, the firm would release the patch to the Open Source community immediately because as time goes on the negative profit (loss) piles on.

Proposition 5: Function p monotonically increases with n if, for all n values, m × Δt ≥ $f(n)$.

Proof: For each increase in n from n to n + 1, we have

$p(n+1) - p(n) =$ m × Δt − $f(n+1) \geq 0$

Under this condition, the firm will never release the patch since the profit generated from the proprietary patch always offsets the integration cost.

Proposition 6: If a n^* value exists such that $f(n^*) < m \times \Delta t < f(n^* + 1)$, $p(n^*)$ is the maximum value of p.

Proof: For every n < n^*, we have

$$p(n^*) - p(n) = (n^* - n) \times m \times \Delta t - \sum_{i=n+1}^{n^*} f(i)$$

That can transform into

$$p(n^*) - p(n) = \sum_{i=n+1}^{n^*} (m \times \Delta t - f(i))$$

Since $f(n)$ is monotonically increasing, we have $f(i) \leq f(n^* + 1) \leq m \times \Delta t$. Therefore $p(n^*) - p(n) \geq 0$ for $n < n^*$.

For every $n > n^*$, we have $p(n^*) - p(n) = (n^* - n) \times m \times \Delta t + \sum_{i=n^*+1}^{n} f(i)$

That can transform into

$$p(n^*) - p(n) = \sum_{i=n^*+1}^{n} (f(i) - m \times \Delta t)$$

Since $f(n)$ is monotonically increasing, we have $f(i) \geq f(n^* + 1) \geq m \times \Delta t$. Therefore $p(n^*) - p(n) \geq 0$ for $n > n^*$.

Since for all $n \neq n^*$ values, we have $p(n^*) - p(n) \geq 0$. Profit $p(n^*)$ is the maximum profit the firm can extract from the proprietary patch. Time t_{n^*} is the optimal time for the firm to release the patch to the community.

4.3. Discussion

From the above analysis, firms make the decision on whether to release their patches back to the community based on the type and state of the Open Source project as well as on how the software is used internally.

If the Open Source software is a widely available commodity (e.g., an operating system) and the patch is not of importance to the firm's core business, the profit cash flow m generated from the proprietary patch might be negligible compared with the re-integration cost in the future. In this case, the firm releases the patch immediately and leverages the community to maintain it for future releases. For example, if the firm finds a flaw in Linux that prevents it to work with some internal applications, the IT staff would develop a patch and submit it to the Linux developer community to be included in the next main Linux release. Once the patch is merged into the main Linux source code repository, other Linux developers will be able to make sure that code changes in future Linux releases will be compatible with the patch. This is the case for most mature level Open Source software.

If new versions of the software containing crucial updates are released frequently (i.e., the Δt is small, the cost to keep the patch updated might be too high and hence trigger the firm to release the patch. This is typically the case for software produced in research projects or in early development stages. In fact, one of the mottos of the Open Source development methodology is to have rapid release cycles. By innovating rapidly, the Open Source developer community could encourage the commercial firms to release their patches earlier in order to keep up with the innovation.

Many Open Source projects started as small projects with limited number of users and simple usage scenarios in mind. As the project grows and gains popularity, the original design often proves inadequate for the scalability requirements. It is common for popular Open Source projects to go through extensive re-architecturing and re-writing several times before it can stabilize. For example, Linux 2.0 is almost a complete re-write from Linux 1.x; The Apache web server 2.0 adopts a different internal architecture from Apache 1.x; Eclipse 3.0 features a different execution kernel and plugin architecture from Eclipse 2.x. For firms that hold proprietary patches, a major re-design of the software could force them to spend significant re-integration costs. A major re-write represents a sharp increase in $f(n + 1)$. According to Proposition 6, it is important for the firm to track the software roadmap and release the patch before the re-write. That allows developers to take into account of the patch in the new design.

5. CONCLUSIONS

In this paper, we analyzed the economic incentives for enterprise software vendors and users to invest in the public good Open Source projects.

For enterprise software vendors, the incentive to invest in Open Source is to create a ”bigger pie” for the commercial externality market. At the growth stage, the firms that have the large market shares in the externality market are the ones most likely to invest. As the project matures, all firms have incentives to invest and the optimal level of investment grows as the market share increases.

For enterprise software users, the incentive to feedback to the community is to avoid the cost of maintaining the patch itself for future software versions. They are most likely to contribute bug fixes for mature commodity software or feature enhancements in frequently updated software.

Chapter 3 Exploring new innovations in mobile computing: applications of wireless technology in financial services, health case and marketing industry

1. SECURE FINANCIAL SERVICES: CHALLENGES AND SOLUTIONS IN THE NEW INFORMATION AGE

1.1.Introduction

The advances of technology have changed how people do business in the past century. Every new invention of manufacturing technology has brought changes to the related industry sector. It is of no surprise that the ongoing revolution of information technology will also revolutionize how we do business in the information intensive financial service industry (Fan et al. 2002). For instance, innovative wireless technologies have been used in financial services and allow users to access market information irrespective of their locations in a timely manner. As we will discuss in the next section, such innovation brings profound changes to the financial markets.

However, innovations in information technologies also pose new challenges. One of the most important challenges in the post-9/11 world is security. Understandably, security concern is even more pungent in financial services. How can we bring convenience and easily accessible financial information to consumers while still ensuring privacy and security? Security technology research in the past several decades have developed a wide variety of effective algorithms and tools. But how to make these technologies suitable for financial market's special needs is a major challenge. For example, building private secular networks for every application may ensure security, but such security model will defeat the very goal of financial services, which is to bring customers access to the financial information from every trading floor of the global markets. Obviously, the demand for high security creates new business opportunities as well as new challenges in today's financial markets. In this article, we address these unique challenges in securing financial services and also suggest several technology solutions to meet these challenges.

In the following sections, we first identify the advances of information technologies in financial markets and discuss the business benefits behind these technical innovations. We then discuss the security priorities and solutions related to those trends. The technology sections of this article are mainly introductory and focused on how various technologies can be used to achieve the security goals we had identified.

1.2. New Information Technologies for Future Financial Services

The Internet has already changed the way most people trade stocks by bringing up-to-date market information and do-it-yourself trading to average investors. As more people directly participate in the market activities, information technology has to evolve to accommodate a variety of new needs. For example, nomadic professional traders want the opportunity to trade when they are on the way; International brokers want to trade in multiple markets simultaneously; Diversified investors want to trade bundled portfolios; High volume traders require high levels of anonymity and security.

To create more secure and flexible financial services to meet these new demands, new technologies are been leveraged. Two technologies that play essential roles are wireless information technology and web services technology. These two technologies offer financial traders unprecedented convenience, choices and speed to access dynamic financial information and to make real-time decisions. This in turn benefits the whole market by increasing liquidity and reducing information asymmetry.

1.2.1. Wireless Information Technology

Wireless information technologies allow traders to access financial services conveniently. Traders can trade at unconventional places and hours. That gives rise to the possibility of 24/7 continuous financial markets, which can handle larger volumes than the current 8 hours a day weekday markets. High volume trading could boost market liquidity and lowers immediacy costs significantly (Fan, et al, 2002, O'Hara, 1997).

Wireless information technologies also allow market information to reach traders more quickly. As Glosten and Milgrom (1985) pointed out, high-speed information flow reduces information asymmetry between knowledgeable traders and market makers, allowing market makers to set more fair bid-ask prices with smaller spreads. It benefits the whole market as liquidity traders will lose less to knowledgeable traders.

1.2.2. Web Services

Financial traders also ask for more dynamic and flexible services from the server side to take full advantage of the increased market opportunities. Those services are likely to come from many competing vendors. One big issue is that how these vendors can talk with each other to provide useful, integrated services to traders. A very promising emerging technology on Internet scale service integration is web services. Web services are self-contained, self-described, dynamically discovered applications with Internet based interfaces. Web services provide platform neutral reusable distributed software components. The key to web services is open standards to achieve interoperability among service providers.

The value of web services technology in financial market is not to establish a monolith service giant. But rather, loosely coupled web services allow a variety of service providers to compete and cooperate in a global financial market. For example, in a stock exchange, the trader needs to go through several steps, including price quote, authentication, order and payment, to complete a transaction. Each of those steps can be handled by web services from competing providers. The traders benefit from the competition among providers at each step.

Web services can talk with each other and dynamically re-configure the behaviors of peer web services. Such automated web services across multiple markets allow traders to access financial information of every global market simultaneously. That helps traders to make informed decisions and capture more trading opportunities. That also helps to increase trade volumes and the speed of information flow, thus increase liquidity of the markets.

However, due to the sensitivity of financial data, any new IT application has to meet very strict security standards. For instance, web services technology champions on open communication standards and allows everyone to compete. But open systems are also more vulnerable to security attacks. Obviously, when new information technologies have radically improved the performance of financial market, they also evoke new security challenges. One key to the success of financial market is to identify and attack security problems. In the next section, we discuss security requirements and special needs for current financial markets.

1.3. Security Challenges

In this section, we will first review general security requirements of financial services. Then, we will discuss special security challenges when we integrate wireless technologies with web services.

1.3.1. General Security Requirements

Security is critical to financial markets. Attacks could selectively disable individual traders or cause denial-of-service (DoS) of the entire market. Attackers could also intercept or modify critical market information. Results of those attacks are severe: they could destroy the trustworthy of the entire financial market system. In general, all financial services, wired or wireless, should meet the following security requirements.

Authentication and Authorization: If a trade execution service receives a trade order from a trader, it must be able to verify the trader's identity and privileges. On the other hand, it is equally important that the trader who receives any financial information can verify that such information is from trusted sources. The authentication check is usually done by verifying digital certificates issued by trusted agencies or by security tokens. The service providers can look up access control directories to determine access privileges for authenticated users.

Prevent Shill Fraud: a trader commits shill fraud if she signs up for multiple identities and poses as several different traders to manipulate the price (Hidvégi, 2002). In interoperable authentication services, shill fraud may be prevented by linking electronic identities with physical identifications (such as driver licenses) or even biometrics information.

Data Integrity: Financial messages have to travel through multiple routers on the open network to reach their destinations. We have to make sure that the information is not modified in the middle of the transmission. The data integrity is usually assured by cross checking the message with a public key encrypted message digest called digital signature.

Confidentiality: Financial transaction data is highly confidential. Even a carelessly leaked intention of large trades could be exploited and causes substantial damage to the trader (see front running attacks below). The only way to ensure confidentiality on a public network is through strong encryption.

Nonrepudiation: When a trader submits an order, she wants to confirm that the broker does receive it. That could save disputation later if the order does not go through for some reason. Nonrepudiation can be guaranteed through a central authority that verifies and time stamps digital signatures

Prevent Denial of Service (DoS) Attacks: DoS attacks flood the financial services or communication channels with enormous amount of useless data and prevent the system from responding to legitimate requests. For investors, a paralyzed financial trading network not only means lost opportunities and lost money, but also directly threatens traders' confidence towards the entire financial market. One especially devastating kind of DoS attacks is clogging attack. Clogging attack exploits the fact that public key operation is very slow (1000 times slower than secret symmetric key operations). The attacker could send a lot of fake public key requests to exhaust the server's computational resources. DoS attacks, especially in their distributed forms, are impossible to completely prevent at the application level. The whole network infrastructure, pricing structure and law enforcement need to be involved to fight against DoS attacks (Geng and Whinston, 2000) .

Prevent Front Running: Front running usually refers to the illegal practice that a broker trades before his clients and makes a profit exploiting the information about his clients' trade intentions. In a broader sense, front-runner can be anyone who can intercept a trader's order information and insert his own trade order before the other trader's. Front running creates a security risk for the financial market since traders can no longer trust that their order information will not be taken advantage of.

1.3.2. Special Security Issues for Wireless Financial Market

Compared with financial market operated in the wired world, wireless financial markets require even more strict security measures because:

Wireless communications are prone to interception. Recent security flaws in IEEE 802.11 standards (Borisov et al. 2001, Miller, 2001) showed that a determined cracker can intercept and decrypt a company's entire local wireless network communications in a matter of hours using a laptop set up in the company's parking lot.

Compared with the wired Internet, the wireless infrastructure is more vulnerable to DoS attacks due to its centralized base stations and limited radio spectrum.

Wireless devices have very limited CPU processing power and are therefore vulnerable to clogging attacks.

Due to the above three weakness and large latency in wireless communications, wireless financial service clients are especially vulnerable to front running attacks.

1.3.3. Special Security Issues for Integrated Financial Markets

Due to the multiple levels of intermediaries involved, security of integrated financial markets powered by web services technologies can be a very complex issue. Traditional Internet secure communication channels may not be sufficient to meet new security requirements.

One crucial component of financial service security is user authentication and authorization. However, multiple vendor supported financial services could make user login a nightmare if each single service provider has its own authentication schemes. That would totally defeat technologies' benefits of convenience. So, single sign-on schemes that are secure and friendly to both service providers and consumers are required.

Although web services can create security challenges, it can also be a powerful tool to provide security solutions. Web services enable us to create security oriented service intermediaries and offer security services as utilities. That could separate the business functions of information service providers from security service providers. It allows users to choose and pay for the security they really need.

In the next section, we survey existing protocols and designs that could meet those security requirements.

1.4. Security Protocols

1.4.1. Secure Communication

On the Internet, communication security is usually ensured by secure sockets based on Secure Sockets Layer (SSL) and Transport Layer Security (TLS) (Dierks and Rescorla, 2002) protocols. In the rest of this article, we refer all SSL/TLS based protocols as SSL for convenience. The secure HTTP protocol (HTTPS) is based on SSL. Since web services communicate with each other using XML-over-HTTP, it is natural to choose SSL to secure financial web services. However, SSL has some serious problems when it comes to meet the security challenges in today's financial market.

SSL is based on point-to-point connection sessions and each SSL session is independent. SSL does not support multiple party or indirect communications very well (IBM, 2002). But in current financial market, each transaction can involve multiple related connections simultaneously to multiple parties and several layers of intermediaries. For example, a stock trade would involve simultaneous connections among brokerage firms, banks, the stock exchange and infrastructure services from the underlying service grid. In addition, many financial information services, such as news services and stock quote services, are based on multicast subscription models. We need to secure the communication content rather than individual SSL connections to provide end-to-end security.

SSL indiscriminately encrypts all communication data using the same key strength regardless of needs. But given the diversity of financial services, some data might need more protection than others. For example, an account number needs stronger encryption than a piece of economic news. Using SSL for such tasks creates unnecessary computational overhead, which makes the system vulnerable to clogging attacks. Also, without a third party server, SSL cannot provide non-repudiation.

To facilitate more flexible security designs in XML applications, several secure XML protocols have been proposed. Secure XML protocols provide ways to transport security meta information with the XML content itself. Most secure XML protocols can easily bind with SOAP messages by adding special XML elements to the SOAP headers. The security information includes digital certificates, security tokens, digital signatures, encrypted data and key references. Those protocols also allow applications to sign or encrypt only parts of the document. Secure XML greatly simplifies the development work to provide efficient end-to-end security through multiple intermediaries and complex network topology. Important secure XML protocols that are being standardized by W3C include XML Digital Signature (DSIG) (Eastlake, et al, 2002a) which provides efficient ways to guarantee data integrity; and XML Encryption (Eastlake, et al, 2002b), which allows encryption of part of the document to provide confidentiality.

On top of those basic protocols, we have to define processes such as the key exchange and policy negotiation. SeXTP (Secure XML Transport Protocol) glues XML DSIG and XML Encryption together to form a client/server communication protocol that could substitute SSL. SeXTP offers a mechanism to support non-repudiation. WS-Security and WS-Trust (Web Services Security and Trust) specifications (IBM, 2002) endorsed by IBM and Microsoft attempt to give a complete solution to web services end-to-end communication security needs.

1.4.2. Authentication and Authorization

As we have discussed, single sign-on is a core security requirement for today’s financial markets. However, each service provider should be able to implement an authentication system that meets its own standards. Then, they can decide to accept authenticated traders from partner realm's authentication servers.

A very promising authentication technology for decentralized single sign-on solutions is Kerberos (Neuman and Ts'o. 1994). Kerberos is based on shared secrets (passwords) between the user and the Authentication Server (AS), and among Kerberos servers in partner realms. The access to a service is granted by a dynamically generated ticket that expires in a short time. Kerberos servers from different realms (brokerage firms, markets, banks and information services) can team up so that they can recognize users authenticated from partner realms. Figure 3.1 illustrates the architecture of Kerberos based financial web services.

In addition to private key based Kerberos, we can also implement single sign-on infrastructure using public key technology. Public key certificates can build trust among parties that have never met before. Digital certificates can be recognized among partner institutions and automatically authenticate certificate holders.

Single sign-on authentication and authorization can also be implemented using standard XML protocols. SAML (Security Assertion Markup Language) (OASIS, 2002) is an XML standard from OASIS (Organization for the Advancement of Structured Information Standards) for exchanging authentication and authorization information. SAML elements can contain Kerberos tickets, security tokens and digital certificates. They can bind with SOAP messages to provide single sign-on services for web services. IBM and Microsoft's WS-Federation, WS-Authorization and WS-Security protocol family (IBM, 2002) promotes a single sign-on scheme competing with SAML.

1.4.3. Wireless Security

Wireless devices could bring traders unprecedented convenience to access opportunities in financial markets. However, wireless devices have very limited computing power to implement sophisticated cryptography algorithms. To enhance overall system security, it is essential to select the right wireless client and network architecture.

On wireless devices, we have two types of client applications to choose from: WAP/WML (Wireless Application Protocol, WAP Markup Language) (Open Mobile Alliance, 2003) browsers and smart independent programs.

WAP is a mature and widely used technology to access the Internet from cell phones and it is already available on millions of phones. Although most WAP browsers can only display WML pages, they could support general XML driven web services through a proxy architecture. An Internet proxy server can interact with web services on behalf of wireless phones. The proxy server keeps track of each wireless phone through cookies. When a phone needs to access one or a combo of web services, it sends the request to the proxy in a pre-agreed WML format. The proxy translates the request to SOAP based service requests to web services. When the web services respond, the proxy combines the results and creates a WML page for the wireless phone. The proxy servers make the WAP based wireless network and XML based web services network transparent to each other (See Figure 3.2). It is relatively cheap for financial institutions to install proxy servers and start to serve their traders on their existing WAP phones right away.

By contrast, smart independent programs on wireless devices can access web services and process XML messages directly. They have rich user interfaces and are easy to customize. Compared with WAP devices, wireless devices running smart programs are bigger and more expensive. However, smart programs can be significantly more secure than WAP applications: (Summarized in Table 3.1)

A smart client can process XML messages directly. Smart clients do not have to rely on the flawed WAP security model (Miller, 2001). They can utilize secure XML protocols to interoperate directly with backend Web services and provide end-to-end security solutions.

A WAP client cannot process application data on its own. All the data storage and processing are done on the server side. WAP based applications are powerless if the network connection is lost. That makes them vulnerable to DoS attacks or even normal traffic congestion. Smart programs allow the wireless device to operate continuously with local data even when the network is temporarily not available (Yuan and Long, 2002).

The lack of ability to process data on WAP devices makes any complex transaction long procedure involving multiple connections, which are prone to errors and security risks (Yuan and Long, 2002). The long latency and multiple connections make WAP clients vulnerable to front running attacks. A properly designed smart program should be able to support atomic transactions.

The WAP proxy architecture relies on centralized proxy servers that are subject to targeted cracking and DoS attacks. If a proxy server is taken out from the network, all the WAP devices it supports become disconnected. By contrast, smart program clients are completely decentralized and have much lower risk.

To summarize, the WAP mobile clients require "always-on" connectivity to function. However, today's wireless networks do not provide that level of reliability and bandwidth. That was a major factor limiting the adoption of WAP-based mobile commerce. As a result, the "occasionally connected" application paradigm supported by smart mobile clients, has become increasingly important for high availability mobile applications.

Smart clients can run standalone on smart devices and only connect to backend data source on an as-needed basis (e.g. daily synchronization). Leading smart mobile client platforms include the Java 2 Micro Edition and Microsoft .NET Compact Framework.

1.5. Conclusion

Financial service is an information intensive industry and it is being revolutionized by the fast advanced of information technology. New technologies such as wireless technology and web services are drastically increasing the trading efficiency for both the markets and individual traders. The improved efficiency can in turn make the market more efficient.

However, development in financial market would be hindered if there is no adequate security measures because the financial market is build upon trust. Wireless web services are still emerging technologies and security issues have not been thoroughly discussed. We identified several core security areas including intermediary assisted communication, single sign-on authentication and authorization, web services security, and wireless clients security. We then surveyed some security solutions including secure XML, Kerberos, smart clients and secure application provision. With industry support on tools and standardization, financial markets will become more secure and benefit every participant in it.

2. USE WIRELESS WEB SERVICES TO IMPROVE PATIENT COMPLIANCE IN CLINICAL DRUG TRIALS

2.1. Patient Compliance and Monitoring in Clinical Drug Trials

Advances in information technology have changed the way researchers and physicians conduct clinical drug trials. Utilizing pervasive computing networks, we are able to put non-intrusive, always-on, network connected information devices everywhere in a patient's life. That could especially benefit outpatient clinical drug trials and therapies conducted in patients' natural settings. Outpatient trials are important because:

They are essential for testing drugs designed to work when the patient interacts with her natural environment. Examples of those drugs include allergy drugs, pain relievers and anti-depression drugs;

Patients do not have to stay in the hospital to get the treatments. That could potentially attract more volunteers to participate in clinical trials. More than 80% of clinical research trials are being delayed because there aren't enough volunteers (Lasalandra, 2002);

Since the trials and therapies are conducted at home, there is no need to use expensive hospital facilities and personnel. The cost can be greatly reduced.

However, compared with inpatient trials, outpatient clinical trials and therapies have to deal with one special issue: the monitoring of patient compliance. Without a controlled environment, it is difficult for the physicians and researchers to know if the patients have followed the trial protocols strictly. In order to evaluate the impact of noncompliance and even proactively reduce noncompliance, we need to monitor the patients' activity. Proper monitoring can also reveal early warning signs to improve the margin of patient safety.

In this article, we will first examine the importance of patient compliance. Then we investigate how wireless electronic data capture (EDC) technology coupled with the wireless Internet and web services could make it easier for both patients to comply with the trial protocols and investigators to monitor the progresses. We will survey current and emerging technology solutions. The technology sections of this article are introductory and focused on how various technologies can help us implement secure and pervasive clinical trial applications.

2.1.1. Compliance to Dosing Instructions

One of the most important aspects of compliance is that the patients need to take the correct dosage of drugs at specified time. Dosing instruction compliance is critical because of the following reasons.

Clinical drug trials are designed to test the safety and efficiency of drugs when investigator's protocols are strictly followed (method effectiveness). The method effectiveness data is later used as the basis to determine appropriate prescriptions for general public patients. However, if a trial patient fails to follow the prescriptions, the individual drug taking pattern would result in a different effectiveness (use effectiveness). Using use effectiveness as method effectiveness causes errors in determining the safe and effective amount of drug prescription for the public.

New drug researches need to collect the accurate drug administration and patient response information to estimate the pharmacokinetic parameters for new drugs (Girard et al. 1996), which indicate how drugs interact with the body in terms of absorption, distribution, metabolism, and excretion. That requires patients to follow the instructions precisely and submit accurate reports on their drug taking patterns.

From the patients' perspective, dosing noncompliance causes undesired outcome and reduces the margin of safety. Missed dosage can change the pharmacokinetics effects of the drug. That is especially a problem for some newly developed drugs which have long dosage intervals designed for convenience (Levy, 1993). Underdose of drugs can cause relapse of serious diseases and development of drug resistance (Kastrissios and Blaschke, 1997). On the other hand, overdose can raise safety risks from side effects.

Another important issue in drug trials and therapies is drug interactions. There are many over-the-counter drugs available in the market. Some of them have undesirable interactions with the others. Since drug interaction is very complex, the knowledge and expertise are generally not available to average patients. It is important to track the exact drugs a patient takes and detect any potential interaction throughout the trials.

Unfortunately, patients often fail to comply with the physician's dosing instructions in outpatient trials and therapies. They easily forget to take drugs on time, especially in the afternoon or evening hours. Missed dosage is the most important drug dosing noncompliance issue in clinical trials and therapies. To make things worse, patients sometimes make up the missed doses later when they remember. Researches have shown that up to 80% to 90% of adult patients fail to take drugs at the frequency prescribed by the physicians. That is even true for patients whose lives are dependent on drug therapies. For instance, studies show that up to 80% of AIDS patients fail to take their drugs on time (Kastrissios et al, 1998)

Various methods have been developed to monitor patients drug taking behaviors in clinical trials. Those methods include patient self reports/diaries, pill counts, special chemical and physiological markers. Based on the drug taking pattern data collected after the trials, statistical models can be developed to correct and calibrate use effectiveness to method effectiveness (Girard, et al. 1998). However, those methods are indirect and only collect/use statistical patterns averaged over a period of time. Thus, they are neither accurate nor suitable for pharmacokinetics studies which require the knowledge of the exact time course of drug exposure and response.

A recent trend is to use EDC devices to monitor the patient's behaviors and record a timestamp for each drug dose (Kastrissios & T. F. Blaschke, 1997). A simple example of EDC is to install a long life battery powered smart chip on the pill bottles. The device is triggered every time the patient opens the bottle. It has proven to be a low cost and accurate patient monitoring mechanism. However, current EDC solutions mainly use simple and standalone devices with little real-time communication and re-configuration capability. They are only passive data collection tools and do not proactively reduce noncompliance. They can not protect patients from underdose and overdose risks. In this article, we will investigate how to take advantage of new wireless technologies and back end information technologies to create dynamic and proactive EDC solutions to not only monitor but also improve patient compliance.

2.1.2. Compliance to symptom report diaries and surveys

It is often important to get both objective data on patient's physical conditions and subjective data on self-reported symptoms during drug trials. Perceived symptom relief reported by the patients is crucial information in evaluating the effectiveness of many drugs. For example, researchers in anti-depression drugs, allergy drugs and pain relievers have long used Daily Symptom Report (DSR) and periodic surveys to evaluate drug effectiveness.

Therefore, it is crucial for the patients to comply with investigator's protocols to report their symptoms accurately and timely. A lot of research efforts have gone into designing the right survey questionnaires that best reflect the patient's perceived symptoms and comfort levels (Wilkie, et al, 2001a). However, these paper and pencil based surveys are intrinsically difficult for patients to comply with. It is cumbersome for the patients to bring the questionnaires along and fill out the forms from time to time. To avoid the trouble, many patients would just fill out the survey at the end of the day. But it is hard to recall accurately the symptom changes throughout the day.

Experiments have been conducted to use interactive computer devices to monitor pain symptoms in hospital settings (Wilkie, et al, 2001b). Through dynamic interactive surveys, physicians can ask patients specific questions about symptoms at real time. Those devices also enable physicians to gather on-time information on early warning signs and thus have great potentials to improve trial and therapy safety. But these computer devices are not suitable for outpatient clinical trials. They are simply too bulky and inconvenient for patients to carry around. We need new tools that are pervasive and can integrate into patients' everyday life to monitor symptom changes.

Appropriately applied in outpatient clinical trials, pervasive EDC devices can monitor both the patients' self-reported symptoms as well as objective vital sign data, such as pulse and blood pressure anytime from anywhere. With smart back end services, EDC information devices can automate and streamline the data flow, improve efficiency and reduce data entry errors. In later sections, we will discuss smart pervasive diary and survey devices in more detail.

2.2. Boost Compliance with Wireless Technologies

2.2.1. Wireless devices as a proactive monitoring tool

In addition to being a monitoring and reporting tool, wireless information devices provide dynamic interaction channels between patients and physicians. Physicians can deliver proactive reminders to patients and thus actively improve compliance rather than merely compensate for the after-fact effects of non-compliance.

Wireless EDC devices can be pervasive in a patient's life: Embedded devices can be attached to the pill bottles; Vital sign monitoring devices can be attached to the body; Diary collecting devices can be carried around as build-in components in cell phones or PDAs. In a wireless clinical trial application, all those EDC devices can be connected through a personal hub device, which could be a TV set-top box at home, an automobile mounted geo-information system in the car or simply an advanced cell phone or PDA with an extra short range communication module. Short range wireless communication technologies such as Bluetooth have made this pervasive personal wireless network possible. Figure 3.3 shows the proposed front end architecture.

If we just want to monitor drug dosing and interactions, the system can be greatly simplified. Instead of wireless electronic chips, we can attach bar code to drug bottles and bar code scanners to hub devices (such as a cell phone). The patient just needs to scan the bottle every time she takes the drug. The patient can also scan the bar codes of other medicines to get instant information about potential interactions with drugs she is taking. The bar code solution could reduce the use of the expensive, complex and potentially insecure local wireless network.

The wireless EDC information hub then communicates with back end services to report data and get further instructions on what to do next. For example, a back end program could contact the hub and instruct it to check the device on the drug bottle at fixed intervals. If the patient missed a dose, the back end service could automatically call the patient's cell phone to remind her. If the investigator decides to change the trial protocol, he could simply update the back end through a web interface. The back end service could then notify the patient and update the monitoring scheme.

2.2.2. Wireless devices as a subjective data collecting tool

Wireless EDC devices can collect more accurate data on the patient symptoms at real time. Since these symptoms are recorded while patients are interacting with their natural environment instead of in the hospital or laboratory, they could better reflect "real life" drug effectiveness. Moreover, wireless EDC devices can improve trial safety by reacting more quickly to help patients when the first sign of abnormal symptom are detected.

Unlike dosing information that could be collected automatically, subjective symptom reports depend on the patients to provide useful data. Pervasive wireless EDC devices provide the patients multiple ways to dynamically input their responses. They could ask the patient to describe symptoms in words, or answer multiple choice survey questions. Those questions need to be interactive and dynamic so as to better fit each patient's unique experiences and environment. The wireless symptom report program could send information to back end services whenever a new symptom develops. If any abnormal symptom occurs, physicians or artificial intelligence programs at the back end could push a real-time survey to the patient and further probe the abnormal situation. They can then make informed decisions on whether any action needs to be taken. If special care is needed, back end services could pull geometric and vital sign data from the personal EDC hub and dispatch required emergency services.

2.2.3. Pervasive user interfaces

One major advantage of using wireless EDC devices is that they cause few disruptions to the patients' normal flow of life. For many users, a small wireless information device can seamlessly melt into their everyday life just like a small notebook or a pen. However, without carefully designed user interfaces, the small size and limited processing power of wireless devices might offset their pervasive advantages. In this section, we will discuss future multimodal mobile applications which combine two types of user interfaces: graphic/text based interfaces (GUIs) and voice based interfaces.

The success of PDAs has demonstrated that desktop GUIs can be migrated to small devices with relatively large screens. Messages and questionnaires to patients can be displayed as text labels or multiple choice boxes on the LCD screens of cell phone and PDAs. The patient can then make choices or enter text in a text form box using a pen-like stylus and/or mini-keyboards. The current hand writing recognition software can run on very small devices and only require minimal training on the user side to write machine recognizable text. For users who are not comfortable with writing on touch screens, they can use portable keyboards. However, GUI is not suitable for all occasions in clinical trials. For example, when a patient needs to fill out dynamically generated symptom report questions, long text input/output on tiny keyboards and small screens can be time-consuming and disruptive.

Compared with writing or typing, speech is probably a more efficient way for the patients to input long answers. A very promising pervasive user interface technology is speech synthesis and recognition. Although on-device real time speech recognition is still beyond the computational capability of the current wireless devices, the situation might change in the next couple years with more advanced algorithms and more powerful wireless chips. Given the current technology, speech based user interfaces can be implemented on the server side. Companies have developed speech based information systems based on VoiceXML (McGlashan et al, 2001). A VoiceXML system has a voice synthesis and recognition engine installed in a VoiceXML gateway and a number of VoiceXML servers. VoiceXML pages themselves are authored in text format and can be dynamically generated from databases. VoiceXML speech recognition and synthesis engines/gateways are usually hosted by an Application Server Provider. The physicians, hospitals and clinical trial operators only need to generate necessary VoiceXML pages.

It is common to setup VoiceXML gateways in call centers and support voice-only applications. However, like any other technology, voice is best fitful for only certain types of applications. For pervasive computing applications in various clinical trials, multimodal applications, which mix voice with graphics, text and even stream video/audio data, are needed. Voice over IP (VoIP) can allow VoiceXML to be fully integrated with graphic/text based user interfaces in a same application. Figure 3.4

illustrates the structure of a wireless multimodal EDC application with integrated voice and digital data user interfaces. Multimodal application is a very active research and development field. Low bandwidth voice data compression algorithms such as IBM's Recognition Compatible Voice Coder (RECOVC), are currently being developed. W3C and the SALT (Speech Application Language Tags) forum are in the processes of standardize markup languages for multimodal applications. So, we anticipate truly operable multimodal mobile applications will be available to clinical trial practitioners in the next several years.

2.3. Back end Services and Infrastructure

2.3.1. Smart back end services

As we have seen from the VoiceXML example, wireless devices do not posses much processing power and they have to delegate a lot of tasks to the back end services. That frees the wireless EDC information hub to do what it is most needed -- to provide pervasive front ends.

The complexity and dynamic nature of drug trial and therapy back end services, as demonstrated in the examples from previous sections, demands a network of services from different providers. Those services include patient recruit and authentication, 24/7 vital sign data monitoring, real time data analysis, proactive warning, emergence services coordination and insurance/payment handling. Those application services are built on the top of infrastructure services such as wireless Internet services, device location services and VoiceXML gateways. All those services can all be modularized into reusable units and outsourced to specialized expert vendors. Figure 3.5 is an illustration of a back end architecture. Many services from many different providers are involved in a heterogeneous network structure.

A serious challenge we face is how to make those service components work together. An emerging technology designed to solve Internet level service integration and interoperation is web services. In the following sections, we introduce web services technology in the context of clinical trial applications.

2.3.2. Interoperable web services

Web services are self-contained, self-described, dynamically discovered, interoperable applications with Internet based interfaces.

Distributed computing services have long been able to work with each other through remote procedure calls and remote object frameworks. However, to build such an interoperable network on the Internet scale requires industry wide standardization of communication protocols. There are XML based protocols to standardize web services' dynamic discovery processes (UDDI), service interfaces (WDSL) and asynchronous and synchronous messaging (SOAP and XML-RPC). There are many other XML protocols to support advanced or industry specific functionalities.

Through UDDI and WSDL, web services can automatically discover and interact with each other without human efforts. Web services communicate with clients and each other over the open Internet through the HTTP protocol. This XML-over-HTTP model can decouple service interfaces by adding a new open, robust, human readable abstraction layer between them. Loosely coupled interfaces are easy to integrate and maintain. Web services technology allows service providers to develop platform neutral reusable distributed software components. The key to web services is open standards and interoperability among service providers.

Under standardized XML interfaces, web services functions are implemented using popular application server technologies. Java and .NET, the two leading server side development platforms, both provide excellent support for web services developments.

On the front end, wireless devices must support web services XML protocols. A very promising wireless development platform is the wireless Java platform. Wireless Java web services APIs are currently being standardized in the Java Community Process (Yuan and Long, 2002).

2.3.3. Web Services Networks

The core concept of web services is to allow multiple vendors to compete in a common market place and therefore give customers the freedom to choose for each service component. To build a complete integrated pervasive computing environment, we also need security, transactional reliability and service level agreements supports.

The future application level web services will build on the top of a network of underlying utility services called the service grid (Hagel and Brown, 2001). Unlike the traditional Internet's connection based architecture, web services are driven by XML messages. Messages can be intercepted and processed by multiple intermediaries along their routes. Those intermediaries are the building blocks of the service grid network. The topology and implementation of the service grid are still in the experimental stage.

One of the most important services that the underlying grid can provide is security. Given the importance of security and trust in health care industry, we will devote the next section to discuss security strategies and how they can be implemented on both the web services end and the wireless end.

2.4. Security

Today's patients are aware of the importance of privacy of their medical records. According to a Gallup pool conducted in 2000, 77% Americans said that the privacy of their personal health information is very important. Yet, according to a 1999 IBM privacy survey, only 23% Americans trust health providers to handle such information properly.

Government regulations such as the Health Insurance Portability and Accountability Act (HIPAA) require health care providers to protect patient's private information. Protecting patient's privacy is equally important in clinical trials. Clinical trial operators and hospitals must provide adequate authentication and authorization checks before anyone can access the patient databases. They also need to guarantee that the data remains confidential when it travels through the open Internet (Hagel and Brown, 2001). In the following sections, we identify security priorities and survey existing security technologies for clinical trial applications.

2.4.1. Authentication and Authorization

Patients and physicians have to be authenticated before they can access sensitive data. After authentication, the system needs to give each individual the appropriate access privileges. For example, a patient is allowed to check in new data and retrieve history data from her own records; An investigator is allowed to monitor and send new instructions to a group of patients participating her trials.

Since most web services are outsourced to independent vendors, it is especially important that all those service providers adopt a uniform single sign-on authentication scheme. Patients and physicians do not have to sign in multiple times to access a variety of integrated services. Furthermore, interoperable single sign-on services present each patient a single point of entry to manage and take full control of all her medical data across many vendors. Advanced forms of authentication services should uniquely identify individuals and automatically authenticate them. For instance, each patient and physician could have a biometric information scanner embedded into their EDC hubs. Such device could provide its owner secure and automatic authentication services.

Due to the multiple vendor nature of the back end web services, it is impractical to let a single company provide centralized authentication services for all parties.

Centralized single sign-on scheme could in fact harm the patients if an untrustworthy monolithic company controls access to all medical data. Therefore, a viable single sign-on model needs to allow each vendor to implement its own security protocols. Then, different vendors can form alliance to accept authentication tokens from partner realms and eventually create an individually controlled, decentralized authentication network. Current security technology such as Kerberos can provide robust implementation for such networks.

2.4.2. End-to-End security through secure XML

Authentication and authorization are only part of the big picture of security issues in pervasive clinical trials supported by wireless web services technology. Communication security is one of the biggest unresolved problems in web services. Unlike traditional COBRA/RMI based remote method calls, web services expose their interfaces out side the corporate firewalls. Moreover, web services are driven by XML messages and the messages have to go through multiple intermediaries in the service grid. For example, a service request to access a patient record has to go through the authentication and authorization services first; A transaction request for insurance co-pay has to be recorded and monitored by the underlying transaction assurance services. Due to the intermediaries involved, traditional Internet secure connections (such as SSL/TLS/HTTPS) can not be used effectively in the world of web services (Yuan et al, 2002). We need a way to secure XML content itself from end to end rather than securing individual intermediate communication channels.

Secure XML standards specify how to embed security information inside XML messages. The security information includes keys, digests, digital signatures, certificates and security tokens. They accompany XML documents from end point to end point.

Secure XML also allows us to encrypt and digitally sign part of the document and therefore implement flexible security policies.

Several secure XML standards have been proposed by Internet standard bodies. W3C's XML Digital Signature and XML Encryption standards provide the basis to store security information in XML documents; OASIS's SAML (Security Assertion Markup Language) (OASIS) defines XML presentation of authentication and authorization information. SAML can contain Kerberos security tokens and be used to support single sign-on web services. Industry leaders such as IBM and Microsoft have teamed up to support a complete family of XML security protocols called WS-Security (IBM, 2002). WS-Security is based on W3C standards but has its own authorization protocol to compete against SAML.

All those secure XML security protocols can bind to SOAP protocols. For example, a SOAP message header can contain an XML Digital Signature segment and a SAML segment to authenticate and authorize itself.

2.4.3. Wireless security

Compared with the wired back end, wireless communications are even more vulnerable to data interception, manipulation and Denial of Service (DoS) attacks. The current wireless communication protocols based on private key algorithms have various security weaknesses. Stronger encryption keys that are longer and slower to compute are needed. Furthermore, if we want to establish trust between parties that have not met before, we need to rely on the very slow (up to 1000 times slower) public key algorithms.

However, wireless devices, especially smart card type devices, have too little battery and processing power to support strong encryption. So, our challenge is to minimize expensive cryptography, especially public key cryptography, operations while still maintaining secure wireless communications. Secure XML standards could be used to meet such challenge. They enable us to implement flexible security policies and use different algorithms and key strengths for different contents.

As has been pointed out by Geng et al in 2000 (Geng and Whinston, 2000), wireless devices are vulnerable to Denial of Service (DoS) attacks. In clinical trails, an attacker can flood the entire wireless spectrum with useless signal and prevent vital sign and other data from reaching the personal wireless EDC hub device. In addition to technological, economical and policy solutions proposed by Geng et al., dedicated radio spectrum can be allocated to medical wireless EDC devices (FCC) to minimize interference from other home wireless devices and thus makes DoS attacks more difficult.

2.5. Conclusions

From our analysis, we conclude that using pervasive wireless information networks to pro-actively monitor patients compliance in drug trials and therapies can bring a win-win situation to both patients and physicians.

The wireless end consists of EDC devices interconnected through a short range wireless peer-to-peer network. The hub device with more powerful CPUs and wireless Internet access communicates with back end services. Wireless EDC devices should provide multimodal user interfaces to the patients.

The back end consists of a network of interoperable web services built around standard XML messaging protocols. Web services networks incorporate a variety of modularized services from competing providers. A major concern for deploying web services is security -- especially in the tightly regulated health care industry. Web services related security technologies have gained important industry support and are developing quickly.

With clear benefits and maturing technology, we might be able to see smart back end powered EDC solutions widely applied in clinical drug trials and therapies in the near future.

3. CALLING ALL CUSTOMERS

No marine biologist would study dolphin behaviors just in a fish tank. Why? Because a fish tank isn't a dolphin's natural habitat, so their behavior in the tank cannot truly reflect their natural behaviors in the ocean. Studying consumer behaviors in a laboratory is like studying dolphins in a fish tank. Laboratory environments are unnatural situations where consumers may behave differently than in the actual shopping and consumption environments.

To better understand the behaviors of any creature, researchers need to conduct field studies in natural habitats. However, compared with laboratory experiments, it's more challenging to do field studies because researchers have to conduct experiments in much larger environments with much less control over contributing factors. How do marine biologists study dolphin behaviors in a habitat as big as the ocean? Sophisticated technologies play an essential role in their research. Electronic tags and hydrophones can help them observe dolphins' natural behaviors in the ocean more accurately.

Human behaviors, however, are more complex. Simple electronic tags that can only track motions not only are intrusive to privacy, but they also offer few insights about human social interaction. More sophisticated methods are clearly needed. For decades, paper and pencil were the main tools for surveying consumer behavior in the field. Although these surveys can be relatively cheap, data coding and entry are notoriously time-consuming and error-prone. These days, researchers use ethnographic research methods to observe consumer behaviors in their day-to-day lives. Although they can provide insights on consumers' natural behaviors that may not be gained from laboratory tests, ethnographic methods are prohibitively expensive and their results are hard to quantify.

Clearly, the lack of advanced and socially acceptable research techniques have severely restrained researchers' ability to study consumer behaviors in their natural environments. However, this situation has begun to change in the past several years. Many promising new technologies are now available to researchers, such as vastly improved wireless data networking technology and advanced database technology. The question is: How do we leverage these technologies to improve consumer research in the field? To address this question, we propose an innovative way to do marketing field research-the wireless marketing survey.

3.1. Innovative Data Collection

The wireless marketing survey is a new data collection technique that conducts marketing surveys on standard wireless phones. In practice, researchers can implement the survey in text-based or voice-based formats or combine both formats in a multi-modal survey. In a text-based survey, a respondent can retrieve a questionnaire and display it as text messages on a cell phone screen. The respondent uses the dial pad to answer survey questions. Then the answers are sent back to the backend database instantly through the wireless communication network and the Internet.

In the voice-based survey, a respondent could listen to the questions from her cell phone and answer the questions by speaking on the phones. All the questions and answers are processed and analyzed automatically by voice synthesis and recognition software on voice extensible markup language (VoiceXML) gateway and servers. The technological implementation of wireless surveys can all be delegated to specialized companies. Market researchers just need to focus on managing the survey process, accessing the survey database, and dynamically interacting with respondents in real time.

Compared with other marketing surveys, the unique charm of wireless marketing survey is its ability to deliver instant answers anytime, anywhere. The pervasive nature of wireless surveys allows marketing researchers to get data from respondents in the natural shopping and consumption environments. Its advantages are numerous:

Contemporaneous. Researchers could use wireless survey to record consumers' real experiences right at the moment of purchase or consumption. To ensure this, researchers could require respondents to fill the survey at a certain moment of consumption or decision making.

Mobile. Like a watch, a wireless phone is always with the consumer wherever they go. It would bring tremendous convenience for the respondents participating in the survey. Other survey instruments, such as computers or wired telephones, don't have this "always with the consumer" feature.

Non-intrusive. Researchers would not ring the respondents' cell phones at random intervals and interrupt the respondents. The respondents have full control of the time and place to fill out the survey. Even when the research requires respondents to fill the survey in certain moments, such requirements are agreed upon by the respondents before the survey and wouldn't be regarded as obtrusive.

Longitudinal. A wireless phone is a personal belonging and can be associated with each individual. This personal feature makes it easy for researchers to keep track of each respondent along a period of time, which could make longitudinal studies easier to implement.

Dynamic. Connecting with the wireless network enables respondents to send answers to the backend database during the survey. The survey can be programmed to branch questions automatically based on the respondent's previous answers.

Geographic sensitive. The geographic location of a wireless phone can be located and tracked in real time. This feature could provide great potential for studying location-sensitive topics, such as a survey conducted in a certain shopping mall or theme park.

3.2. Potential Applications

Wireless marketing surveys can be used in numerous marketing studies. For example, Ju Long, Michael Yuan, and colleagues have studied how to use wireless devices to increase patient compliance in a clinical drug trial, where real-time reporting of patient symptom changes and drug effects from any location is critical.

The ability to obtain this type of time-sensitive information is one of the key benefits of wireless surveys. Literature on human cognition and memory has shown that data collected immediately is more accurate than data collected through retrospective methods because individuals' memories on feelings and events are often biased and distorted. For instance, Schwarz and Sudman (1994) suggest that, when subjects are asked to summarize experience over a time interval, more recent experiences have a greater influence on recall than more distant ones. Such distortion could create erroneous results for marketing research. For instance, if we ask a consumer to recall her experience on a cruise trip to Caribbean, the consumer may only recall those events that happened at the end of the trip while forgetting events at the beginning of the trip. However, if the respondent could fill out a wireless survey at the moment of each event, researchers could have gained a much more accurate understanding of the customer's experiences.

Moment of purchase. How consumers make their decisions at the moment of purchase still remains a mystery to marketers. Most data gathered from the actual shopping environment, such as scanner data, are post-purchase data. It only shows what consumers buy, but not why. This "black-box" perspective is persistent because researchers didn't have a way to study consumer motivation at the moment of purchase.

Wireless marketing surveys could help researchers open the black-box and better understand factors that influence consumers' purchase decisions, especially the influences of situational factors.

As defined by Russell W. Belk (1975), situational factors are factors particular to a time and place and have a demonstrable and systematic effect on current consumer behavior. Situational factors' influences are common in consumer decisions. For example, time since last meal could significantly affect a consumer's purchase decision between a half-gallon ice cream and a one-pint ice cream.

To measure the influence of situational factors, survey instruments must be available to respondents in those specific situations. That's where wireless surveys can be valuable. For instance, marketers could use wireless surveys to study the effect of background music on buyers' lingering time at a store or how the amount of money available at the moment of shopping may affect consumers' impulsive buying behavior.

Affective factors of buyer behavior. According to the affective-as-information framework proposed by Norbert Schwarz (1990), evaluation of the product is not just a cold, reasoned assessment. In fact, consumers rely heavily on the "how do I feel about it?" heuristic to make choices. For instance, consumers buy mobile phones not only because they're useful, but also because they feel cool. The success of Nokia's fashion designed handset clearly demonstrates the heuristic factors in consumer choices.

Although affective heuristic factors are prevalent in consumer behaviors, measuring it is difficult. Affective factors are temporal and constructed on the spot. Hence they are highly susceptible to the context and difficult to recall. Most field surveys can only rely on retrospective measures, which are often erroneous.

To address such problems, Arthur Stone and colleagues (1999) propose using moment-to-moment measures and eliminate recall bias by using immediate self-reports.

Wireless surveys exactly fit this proposition. With wireless phone in hand, respondents don't have to rely on their memories to recall their feelings. For example, marketers can use wireless surveys to test how a prospective car buyer feels about the car during the test drive and get the fresh and accurate information on the spot.

Satisfaction at time of consumption. The ultimate goal of marketing professionals is to make customers happy and satisfied. But is the measurement of customer satisfaction accurate? Does the data truly reflect the consumer's experiences? Intuitively, data collected at the moment when the consumers are using the products/services can more accurately show whether consumers are satisfied or not. Wireless surveys can be used to do this kind of study. For instance, if you want to know how consumers enjoyed their visits to Disneyland, you can let selected visitors fill out wireless surveys while they tour around the park.

In addition, wireless survey can be used to examine the affective factors in customer satisfaction. Richard Oliver (1993) suggests that disconfirmation of expectations can't adequately explain the formation of customer satisfaction. Affective factors during the consumption experience play an equally important role in customer satisfaction. For example, a new car owner can be satisfied with her car because it feels trendy even if its gas mileage didn't meet her expectation. Using wireless surveys, marketing researchers can measure these affective factors more accurately and better understand consumers' satisfaction formation process.

Longitudinal consumer behavior. Consumer behavior isn't made up of static phenomena and lone events. Instead, it's a continuous and dynamic process covering the entire purchasing and consumption time span. Brand loyalty, for example, is not built in a flash, but arises from trust gained over many transactions. So it's important to integrate time as a dimension in consumer researches.

However, few consumer behavior studies have adopted longitudinal designs. Most studies are static and one-shot because keeping track of respondents is a tedious and costly task. To conduct longitudinal studies on consumer behavior, marketers need a suitable technique that can track respondents in a timely and non-costly manner. Wireless surveys are useful in implementing longitudinal studies. The "always on" feature makes it easy for researchers to keep in touch with the respondents over time.

Interpersonal influence on consumer behavior. The magic power of word of mouth has long been treasured by marketing professionals. One word from mom is far more trustworthy than a thousand advertisements. But little is known on how it works. For instance, does it have a different influence on men and women? How does it affect customer satisfaction? Answers to these questions will help marketers better understand and leverage the power of word-of-mouth in marketing campaigns.

During wireless surveys, researchers can initiate wireless communication between the respondents through voice- or text-based messaging and study how this interpersonal communication influences consumers' behavior.

Effectiveness of advertisements. The goal of advertising is to have a long-term impact on consumers, which helps build brand equity. Longitudinal wireless surveys can be a good way to study advertisements' long-term effectiveness.

Furthermore, wireless surveys can be used to study effectiveness of outdoor advertising, a long overlooked area in marketing research. According to the Outdoor Advertising Association of America (OAAA), in 2001, advertisers spent a total of $5.3 billion on outdoor media. Surprisingly, little research has been done on this billion-dollar market. How effective are those advertisements on highway billboards? Wireless surveys can help answer those questions because the wireless phone is always available to respondents on the road.

3.3. Implementation Procedure

Researchers and survey administration software. Wireless surveys fully leverage the power of the Internet. Using Web-based survey management software, researchers can design the survey questionnaire, monitor survey processes, and interact with respondents from anywhere convenient to them. Data also can be transferred, stored, and analyzed immediately from the Web.

Respondents. All it takes to participate in a survey is a standard Internet-enabled cell phone. Respondents can either be selected from a consumer panel or sampled from a larger population. Some basic training may be needed if the respondents don't have previous experiences using the wireless Internet. Because all wireless phones have embedded IDs that can't be tampered with, wireless surveys could support automatic user authentication to prevent fraud. Standardized IDs also make it easy to track the respondent's long-term behavior.

Application servers and wireless access provisions. The backend application server processes all the communications between researchers and respondents, stores data, and executes commands from survey administration software.

Research companies can either host the application server by themselves or outsource the hosting tasks to an application service provider (ASP). Hosting the server can give researchers more control over the applications. A wireless data collection platform needs to be installed on the server. On the other hand, outsourcing to the ASP will let the ASP take care of all the technological details. It can provide a turnkey package, including platform development, application development, hosting server, and wireless access provision to its clients. The researchers only need to know how to use the survey management software.

3.4. Challenges.

Due to the limited display spaces on the cell phone screens, simple and short questions are preferred in text-based wireless surveys. In the voice-based survey, the same rule of thumb also applies. Hence, wireless surveys may not be suitable for research that involves long questions and answers. Cell phones have limited capacity to handle graphics, so survey questions should mainly be text-based.

3.5. Endless Opportunities

More companies are starting to realize they can gain a competitive edge by making sure their products/services fit in consumers' lives. The value of wireless marketing surveys lies in their ability to develop a rich understanding of consumer natural behaviors in daily lives. Compared with traditional tools, they can provide marketing researchers with more accurate and timely information on when, where, and what consumers buy-and, more important, why they buy. Leveraging these new insights, marketing professionals can develop more effective marketing strategies for product development, distribution, pricing, and promotion.

Wireless marketing survey is a brand-new data collection method. With the recent vast advances in the wireless Internet technologies and VoiceXML technologies, and the exponentially growing wireless consumer market that provides an expanding respondent base, we believe the opportunities in wireless marketing survey are unlimited.

Appendix for Chapter 1

	Forum	Download	Bug	Patch	CVS	Release	News
N Valid	151	291	299	299	293	273	220
N Missing	149	9	1	1	7	27	80
Mean	577.54	355017.53	212.97	34.02	4071.81	18.34	10.39
Median	114	24545	49	3	1413	12	6
S.D.	1378.12	1839506.96	650.62	189.38	11014.05	21.03	14.77
Skewness	4.39	10.16	7.34	12.12	8.32	2.55	4.24

Table 1.1: Frequency table before data transformation

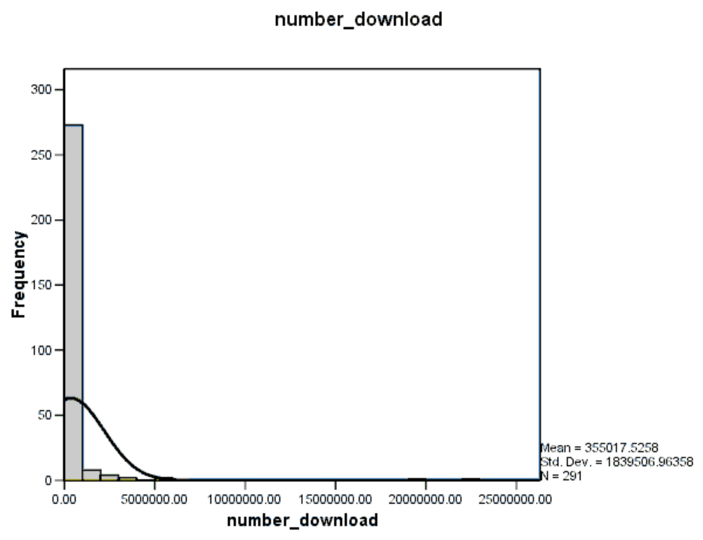

Figure 1.1: Histogram of number of download before log-transformation

	Forum	Download	Bug	Patch	CVS	Release	News
N Valid	151	287	290	197	288	272	218
N Missing	149	13	10	103	12	28	82
Mean	4.79	10.32	4	2.28	7.2	2.37	1.82
Median	4.74	10.16	4	2.2	7.29	2.48	1.79
S.D.	1.81	2.12	1.63	1.63	1.54	1.1	1
Skewness	0.21	0.23	0.92	0.51	-0.27	-0.26	0.22

Table 1.2: Frequency table after data transformation.

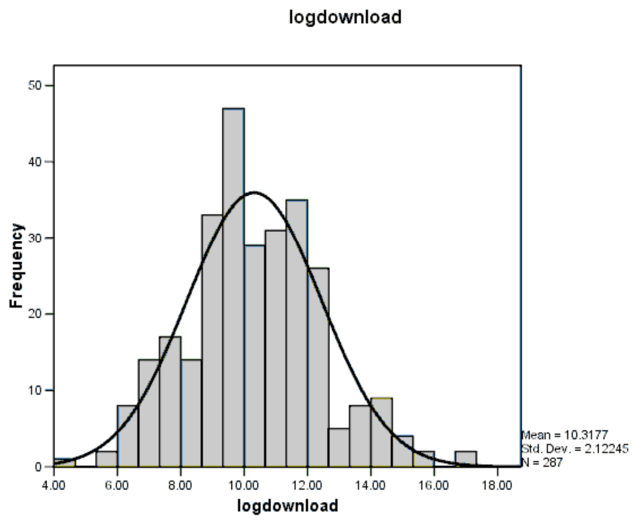

Figure 1.2: Histogram of number of download after log-transformation

Component	Initial Eigenvalues			Rotation Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	3.633	36.330	36.330	2.205	22.047	22.047
2	1.245	12.452	48.781	2.143	21.432	43.479
3	1.113	11.126	59.908	1.262	12.624	56.104
4	.933	9.327	69.235	1.122	11.220	67.324
5	.849	8.493	77.728	1.040	10.404	77.728
6	.700	7.003	84.731			
7	.557	5.572	90.303			
8	.387	3.866	94.169			
9	.359	3.590	97.759			
10	.224	2.241	100.000			

Table 1.3: Extraction communalities table

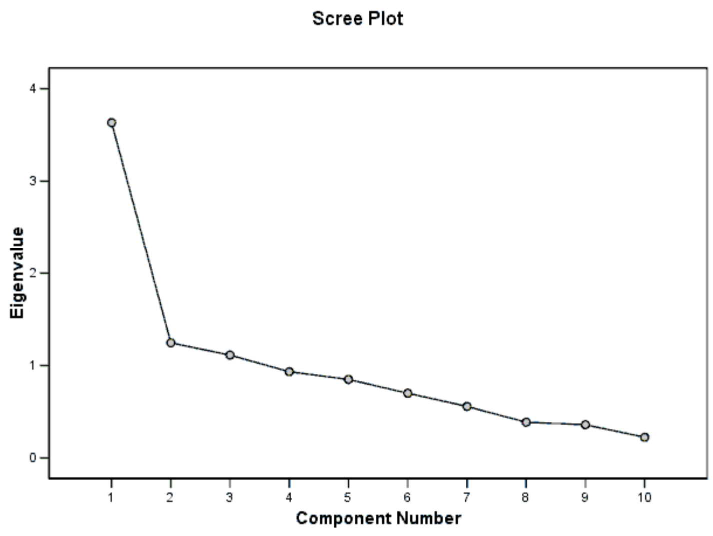

Figure 1.3: Scree plot

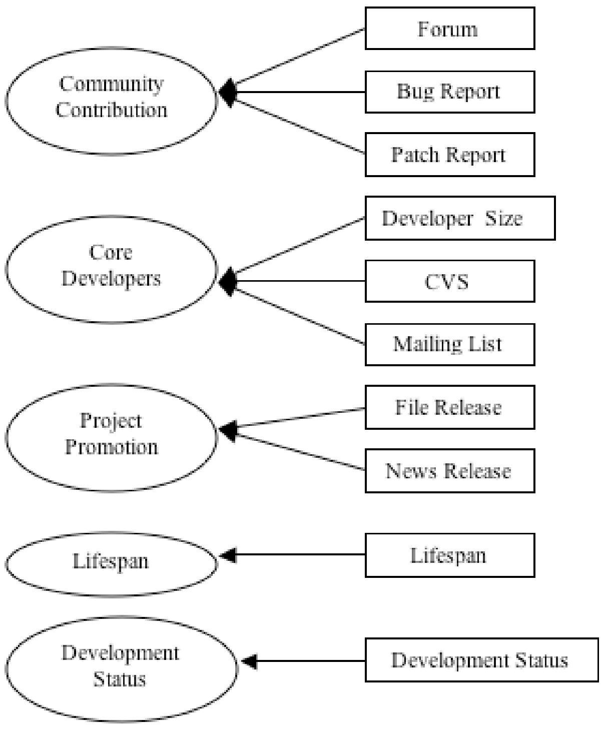

Figure 1.4: Latent variables and manifest variables

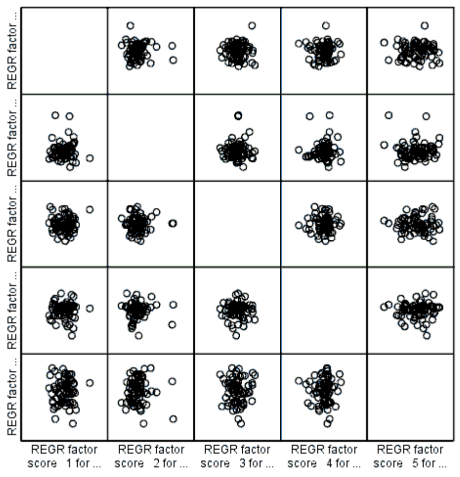

Figure 1.5: Scatter plot matrix of the component scores

Mode l	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.682(a)	.465	.423	1.53624

Table 1.4: ANOVA

Mode l		Sum of Squares	df	Mean Square	F	Sig.
1	Regressio n	131.077	5	26.215	11.108	.000(a)
	Residual	151.042	64	2.360		
	Total	282.119	69			

Table 1.5: ANOVA 2

Model		Unstandardized Coefficients	Standardized Coefficients		T	Sig.
		B	Std. Error	Beta		
1	(Constant)	11.071	.184		60.295	.000
	Community Contribution	.828	.185	.410	4.480	.000
	Core Developers	.745	.185	.368	4.027	.000
	Project Promotion	.620	.185	.306	3.350	.001
	Project Lifespan	.257	.185	.127	1.391	.169
	Development Status	.457	.185	.226	2.469	.016

Table 1.6: Coefficient of ANOVA

Model		Correlations			Collinearity Statistics	
		Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)					
	Community Contribution	.410	.489	.410	1.000	1.000
	Core Developers	.368	.450	.368	1.000	1.000
	Project Promotion	.306	.386	.306	1.000	1.000
	Project Lifespan	.127	.171	.127	1.000	1.000
	Development Status	.226	.295	.226	1.000	1.000

Table 1.7: Collinearity test

Appendix for Chapter 3

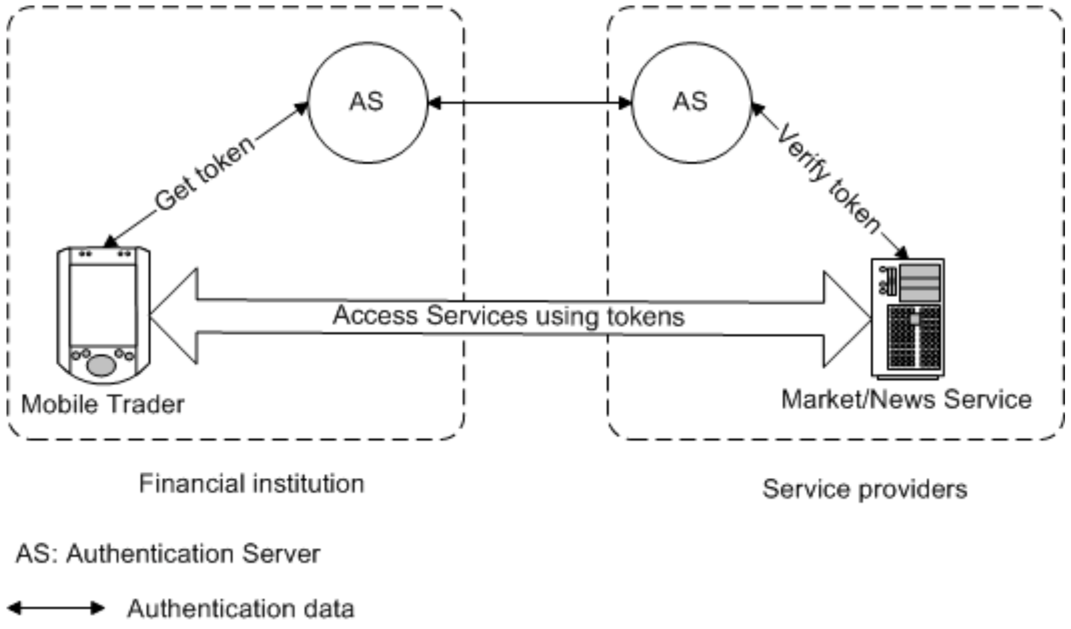

Figure 3.1. Kerberos based financial services market. Financial services are divided into security realms.

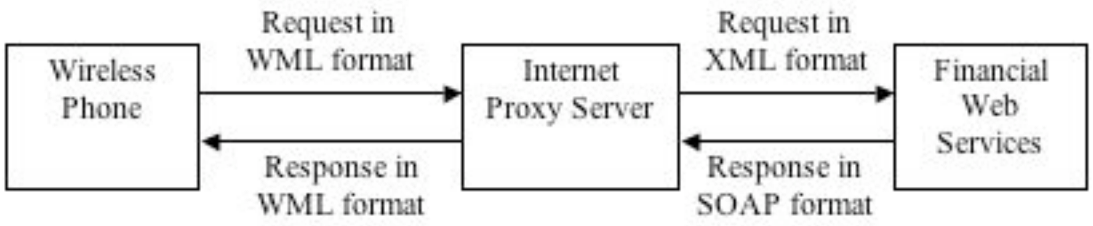

Figure 3.2: Proxy servers make the WAP based wireless network and XML based web services network transparent to each other.

Smart Client	WAP Client
Can process secure XML Enable end to end security solutions	Have to rely on proxy
Use local process power	Cannot process data on its own,
Can work continuously	Vulnerable to DoS attacks and traffic congestion
Support atomic transaction	Multiple round-trips cause potential data corruption and long latency
Decentralized	Centralized proxy servers
With lower risk	Subject to DoS attacks and other attacks

Table 3.1: Comparison between Smart Client and WAP Client

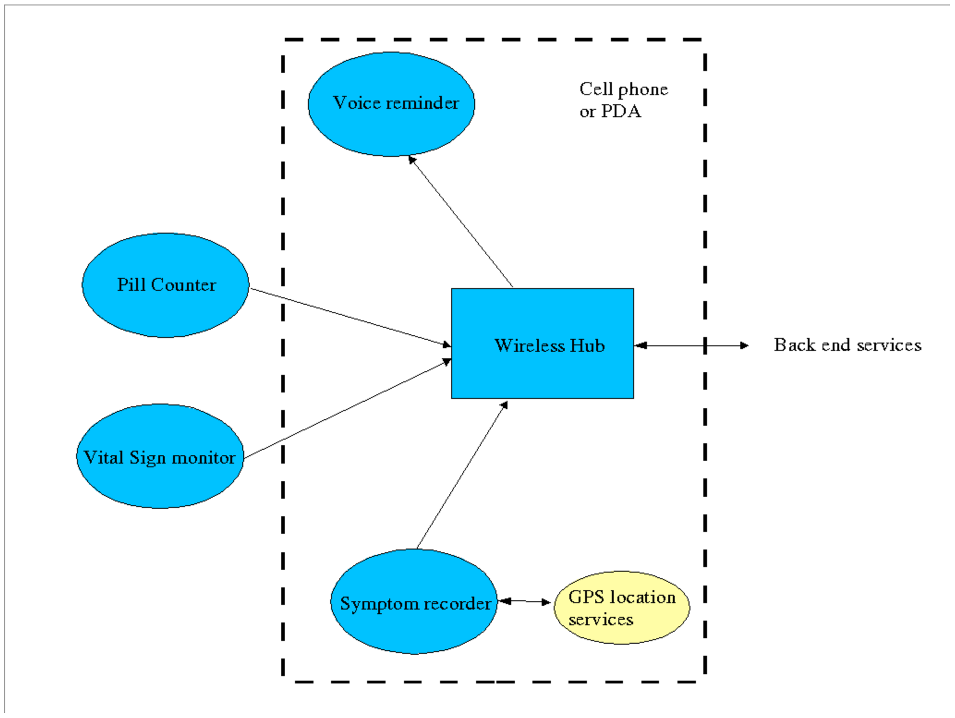

Figure 3.3: The wireless front end. Components in the dashed line box can be separate devices or combined into one single hub device.

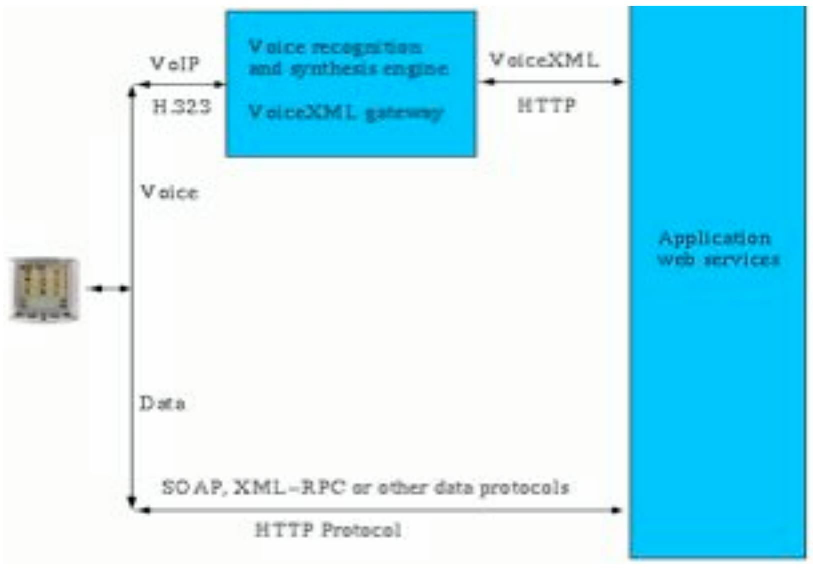

Figure 3.4: Mix voice and graphics user interfaces in the same multimodal wireless application.

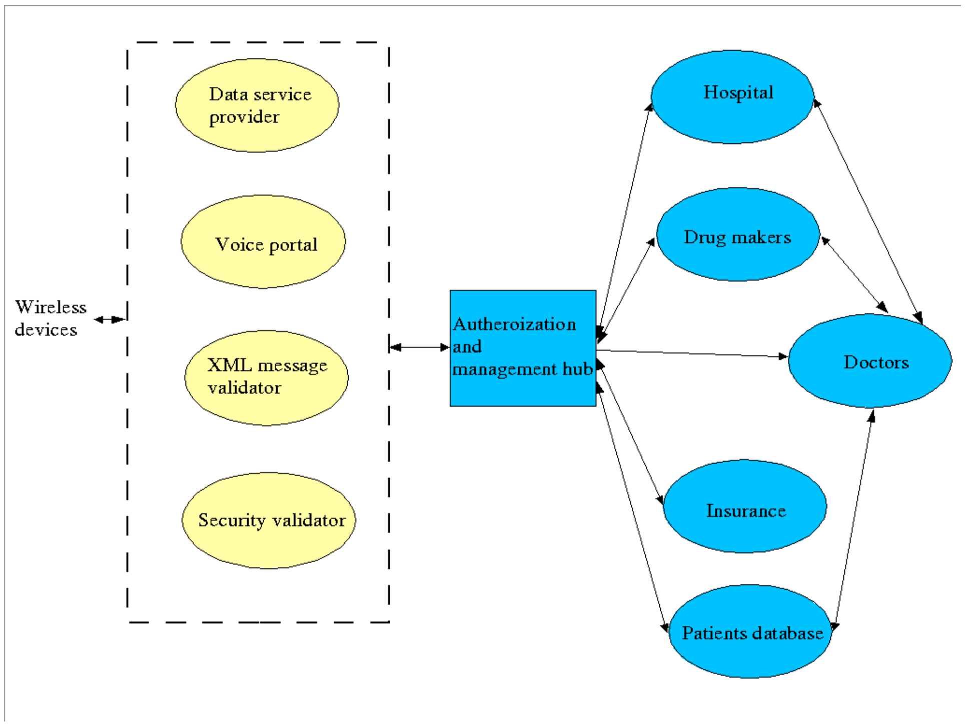

Figure 3.5: The complex web services architecture at the back end.

References

Borisov, N., Goldberg, I. and Wagner, D. 2001. “Intercepting Mobile Communications: The Insecurity of 802.11”. *Proc. 7th Ann. Intl Conf. Mobile Computing and Networking.*

Miller, S. K. 2001. “Facing the Challenge of Wireless Security”. *IEEE Computer*, July, 34 (7), 16-18.

Neuman, B. C. and Ts'o, T. 1994 “Kerberos: An Authentication Service for Computer Networks”. *IEEE Communications*, 32 (9), 33-38.

OASIS. 2002a. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). Committee Specification.

Stanford, V. 2002. "Pervasive Health Care Applications Face Tough Security Challenges", *IEEE Pervasive Computing*, Vol. 1, No. 2, 8-12.

Yuan, M.J., Long, J. & Whinston, A.B. 2002, "Secure Financial Web Services for Wireless Clients", *International Journal of Electronic Commerce*, under review

O'hara, M. 1997. Market Microstructure Theory. *Blackwell Publishers.*

OASIS. 2002b. "XML-Based Security Services TC (SSTC) Security Assertion Markup Language".

IBM Corporation and Microsoft Corporation, 2002. "Security in a Web Services World: A Proposed Architecture and Roadmap", *White Paper*, April 7, 2002 Version 1.0

Yuan, M. J. and Long, J. 2002a. Securing Wireless J2ME. *IBM developerWorks*,

Geng, X. and Whinston, A.B. 2000. "Defeating Distributed Denial of Service Attacks", *IEEE IT Professional*, Vol. 2, No. 4, page 36.

Yuan, M. J. and Long, J. 2002b Java readies itself for wireless Web services. *JavaWorld*,

Federal Communication Commission, 2000. "Amendment of Parts 2 and 95 of the Commission's Rules to Create a Wireless Medical Telemetry Service", *Report and Order*, FCC 00-211.

Dierks, T. and Rescorla, 2002. E. The TLS Protocol. *IETF Internet Draft.*

Girard, P., Sheiner, L.B., Kastrissios, H. & Blaschke, T.F. 1996. "Do we need full compliance data for population pharmacokinetic analysis?" *J. Pharmacokin. Biopharm.* Vol 24, pp265-282.

Eastlake, D. 2002a. XML-Signature Syntax and Processing. *W3C Recommendation*

Levy, G. 1993. "A Pharmacokinetic perspective on medicament noncompliance" *Clin. Pharmacol. Ther.* Vol 54, pp242-44.

Eastlake, D. 2002b. XML Encryption Syntax and Processing. *W3C Candidate Recommendation*

Kastrissios, H. & Blaschke, T.F. 1997. "Medication compliance as a feature in drug development" *Annu Rev Pharmacol Toxicol.* Vol 37, pp451-75.

Fan, M., Srinivasan, S., Stallaert, J. and Whinston, A. B. 2002. "Electronic Commerce and the Revolution in Financial Markets". *Thomson Learning.*

Kastrissios, H., Suarez, J.R., Girard, P. Sheiner, L.B. and Blaschke, T.F. 1998. "Characterizing patterns of drug-taking behavior with a multiple drug regimen in an AIDS clinical trial." *AIDS,* vol 12, pp2295-2303

Geng, X. and Whinston, A. B. 2000. Defeating Distributed Denial of Service Attacks. *IEEE IT Professional*, 2 (4), 36-42.

Girard, P., Blaschke, T.F., Kastrissios, H. and Sheiner, L.B. 1998. "A Markov mixed effect regression model for drug compliance." *Statist. Med.* vol 17, pp2313-2333.

Wilkie, D.J., Huang, H.Y., Reilly, N. and Cain, K.C. 2001. "Nociceptive and neuropathic pain in patients with lung cancer: a comparison of pain quality descriptors" *J. Pain Symptom Manage*. vol 22, pp899-910.

Glosten, L. and Milgrom, P. Bid, 1985. Ask and Transaction Prices in a Specialist Market with Heterogeneously Informed Traders. *Journal of Financial Economics*, 14, 71.

Wilkie D.J. 2001, "Cancer symptom control: feasibility of a tailored, interactive computerized program for patients" *Fam Community Health.* vol 24, 48-62.

Hidvégi, Z., Wang, W. and Whinston, A. B. 2002. Sequentially Optimal Auctions under Shill Bidding. *CREC Working Paper.*

IBM Corporation and Microsoft Corporation. 2002. Security in a Web Services World: A Proposed Architecture and Roadmap. *IBM developerWorks.*

McGlashan, S. et al. 2001. "Voice Extensible Markup Language (VoiceXML) Version 2.0", *W3C working draft*.

Asundi, J., 2001. Software engineering lessons from open source projects. In Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds).

Belk, Russell. W., 1975, "Situational Variables and Consumer Research," Journal of Consumer Research, Vol. 2, 157-164.

Fielding, R.Y., 1999. Shared leadership in the Apache project. Communications of the ACM, 42 (4), 42– 43.

Free/Libre and Open Source Software: Survey and Study, International Institute of Infonomics, University of Maastricht, The Netherland; Berlecon Research GmbH, Berlin, Germany, June 2002, http://www.infonomics.nl/FLOSS/report/

Hars and Or, 2002, “Working for Free? Motivation for Participating in Open Source Projects”, International Journal of Electronic Commerce, Vol. 6, No. 3.

Healy and Schussman, 2003, University of Arizona, Department of Sociology, working paper

Ioannis S, Ioannis S, Lefteris A, and Apostolos O, 2004, “Open Source Software Development Should Strive for Even Greater Code Maintainability”, Communications of the ACM, October, pp. 83-87.

Koch, S; Schneider, G, Effort, co-operation and co-ordination in an open source software project: GNOME. Information Systems Journal, Jan2002, Vol. 12 Issue 1, p27, 16p [used CVS and forum as electronic communication artifact to show project development]

Lerner, J, Tirole, J. 2002. “Some Simple Economic of Open Source”, Journal of Industrial Economics, 50, 197-234.

March, J.G. & Simon, H.A., 1958. Organizations. John Wiley, New York.

Markus, M.L., Manville, B. & Agres, C.E. 2000. What makes a virtual organization work? Sloan Management Review, Fall, 13–26.

Mintzberg, H. 1972. The Structure of Organizations. Prentice Hall, Englewood Cliffs, NJ.

Mockus, A., Fielding, R.T. & Herbsleb, J. 2000. A case study of open source software development: the Apache server. Proceedings of the 22nd International Conference on Software Engineering, 263–279

Moody, G. 2001. Rebel Code: Linux and the Open Source Revolution. Perseus Press Cambridge, MA.

Nohria, N. 1995. Note on organization structure. Harvard Business School, Reprint no. 9–491–083.

Oliver, R. 1993. "Cognitive, Affective, and Attribute Bases of the Satisfaction Response", Journal of Consumer Research, Vol. 20, Issue 3, 418-431.

Raymond, E. 1999. The Cathedral and the Bazaar.

Schmidt, D.C. and Porter, A. 2001. Leveraging open source communities to improve the quality and performance of open source software. In: Making Sense of the Bazaar: Proceedings of the 1st Workshop on Open Source Software Engineering. Feller, J., Fitzgerald, B. & van der Hoek, A. (eds).

Schwarz, N. 1990, "Feelings as Information: Informational and Motivational Functions of Affective States", in E. Tory Higgins and Richard M. Sorrentino (Eds.), Handbook of Motivation and Cognition, Vol. 2, New York: Guilford, 527-561.

Schwarz, N. and Sudman, S. 1994. Autobiographical Memory and the Validity of Retrospective Reports, New York: Springer-Verlag.

Sharma, S. Sugumaran, V. Rajagopalan, B, 2002. A framework for creating hybrid-open source software communities. Information Systems Journal, Vol. 12 Issue 1, p7-26

Stone, A, Shiffman, S. and DeVries, M.W. 1999. "Ecological Momentary Assessment", In D. Kahneman, E. Diener, & N. Schwarz (Eds.), Well-being: Foundations of Hedonic Psychology, New York: Russell-Sage.

Thomas and hunt, 2004, Open Source Ecosystems, IEEE Software 32(1)

Hagel, J. and Brown, J.S, 2001 October. "Your Next IT Strategy", *Harvard Business Review*, page 105.

Lasalandra, M. April 14, 2002. “Docs calling all volunteers - Shrinking test pool delays new drug trials”. *Boston Herald,* page 9.

Yuan, M.J. and Long, J. 2002 June. "Java readies itself for wireless Web services", *JavaWorld.*

Vita

Ju Long was born in Chengdu, Sichuan, People's Republic of China on July 11, 1974, the daughter of Bangyi Liu and Yongtao Long. After completing her work at Shishi High School, Chengdu, Sichuan, in 1992, she entered Renmin University in Beijing, China. She received the degree of Bachelor of Arts from Renmin University in May 1996. She then entered the Graduate School of Renmin University and received the degree of Master of Business Administration from Renmin University in May 1999. From 1998 to 2000, she entered the Graduate School of The University of Michigan in Ann Arbor and received the degree of Master of Social Work in May 2000. In September 2000, she entered the Graduate School of The University of Texas at Austin.

Permanent address: Apartment K, 1628 W 6th Street, Austin, Texas 78703

This dissertation was typed by Ju Long.