Framework for Analyzing the Success of Open Source Software

Thesis

Presented in Partial Fulfillment of the Requirements for the Degree
Master of Science in the Graduate School of
The Ohio State University

By

Arun Chockalingam

Graduate Program in Computer Science and Engineering
The Ohio State University

2011

Thesis Committee:

Dr. Rajiv Ramnath, Advisor

Dr. Jay Ramanathan
ABSTRACT

The open source industry has grown by leaps and bounds over the past decade. The concept of open source has been beneficial to a lot of people and has also proven to be a learning platform. It has forced the commercial organizations to change their business models. However, while some open source software has been enormously successful, most of it fails. An estimate is that almost 90% of all Open Source software fails.

This has led to a lot of research on the factors that contribute to the factors of open source Software (OSS). Most of the existing research has been exploratory or case study based in nature, while some of it has been empirical. The majority of existing work has focused on one particular factor for OSS success, or has been a case study of one or two particular projects. Even though there has been a lot of research in this topic, there is no clear method described on how a project administrator or sponsor could use this information to analyze his own project and improve it.

We address this need by creating a framework of measures that can be used to analyze the success of any OSS project. This can be used by prospective and current developers and sponsors to analyze their project and improve it based on the results, to make it successful. We also provide guidelines on collecting data for applying the framework. Finally,
we also apply the framework on two case study projects, namely, KANSEI and JFtp, discuss the results of the analysis, and suggest areas of improvements. We also perform a preliminary analysis of a third project, namely the On-time Measure project and discuss the results of the analysis.
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the best advisor and mentor I could have possibly got, Dr. Rajiv Ramnath. His knowledge and critical thinking has been very valuable. His encouragement and confidence has made this thesis possible.

I would also like to thank Professor Jayashree Ramanathan and Dr. Thomas Bitterman for their continued support, insights and suggestions that helped shape my research.

I also really appreciate the support I got from my family, especially my mother, and my close circle of friends who kept encouraging me constantly and kept me on track to complete this successfully.
VITA

May 2009……………………… B.Tech. Information Technology,
Anna University

September 2009 to present......Graduate Student, Department of Computer
Science and Engineering, The Ohio State
University.

Field of Study:

Major Field: Computer Science and Engineering
# TABLE OF CONTENTS

ABSTRACT .................................................................................................................. ii

ACKNOWLEDGEMENTS .............................................................................................. iv

VITA ............................................................................................................................. v

TABLE OF CONTENTS ............................................................................................... vi

LIST OF FIGURES ........................................................................................................ xi

1. INTRODUCTION ...................................................................................................... 1

   1.1 The Problem: ......................................................................................................... 6

   1.2 Solution Approach ................................................................................................. 8

2. RELATED RESEARCH ............................................................................................. 10

   2.1 Motivation for OSS Developers ........................................................................... 12

   2.2 License Types ....................................................................................................... 14

   2.3 Network Characteristics ...................................................................................... 16

       2.3.1 Network Size ............................................................................................... 16

       2.3.2 Network Density .......................................................................................... 16
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2 Network Diversity</td>
<td>17</td>
</tr>
<tr>
<td>2.3.3 Project and Developer Degree Centrality</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Software Quality</td>
<td>17</td>
</tr>
<tr>
<td>2.4.1 Software Development practices</td>
<td>18</td>
</tr>
<tr>
<td>2.5 Team Characteristics</td>
<td>19</td>
</tr>
<tr>
<td>2.5.1 Trust</td>
<td>19</td>
</tr>
<tr>
<td>2.5.2 Shared Understanding</td>
<td>20</td>
</tr>
<tr>
<td>2.6 Success Measures</td>
<td>21</td>
</tr>
<tr>
<td>3. THE FRAMEWORK</td>
<td>28</td>
</tr>
<tr>
<td>3.1 Domain Size and Competition</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Development Status</td>
<td>31</td>
</tr>
<tr>
<td>3.3 Team Structure</td>
<td>32</td>
</tr>
<tr>
<td>3.4 Usage</td>
<td>34</td>
</tr>
<tr>
<td>3.5 License:</td>
<td>35</td>
</tr>
<tr>
<td>3.6 Programming Languages Used</td>
<td>37</td>
</tr>
<tr>
<td>3.7 Portability</td>
<td>39</td>
</tr>
<tr>
<td>3.8 Developer Satisfaction</td>
<td>40</td>
</tr>
</tbody>
</table>
3.9 User Satisfaction ................................................................. 41
3.10 User Participation .............................................................. 42
3.11 Organizational Sponsor .......................................................... 44
3.12 Reputation and Recognition ..................................................... 46

4. DATA COLLECTION METHODS .................................................. 49

4.1 Case Study Projects ............................................................... 51
  4.1.1 Jftp .............................................................. 51
  4.1.2 KANSEI ........................................................... 51
  4.1.3 On-time Measure .......................................................... 52

4.2 Surveys .................................................................................. 53
  4.2.1 KANSEI User Survey: ...................................................... 53
  4.2.2 KANSEI Developer Survey: ............................................. 56
  4.2.3 Jftp Developer Survey: ................................................... 58
  4.2.4 Jftp User Survey: ........................................................... 60

5. ANALYSIS OF THE CASE STUDIES AND ITS RESULTS .......... 62

5.1 Application of the Framework ................................................... 62

5.2 Discussion of the Analysis Results of KANSEI and JFtp ........ 69
5.3 Discussion of the preliminary analysis of On-time Measure ...... 70

6. CONCLUSIONS AND FUTURE WORK .................................. 71

REFERENCES........................................................................ 73

APPENDIX A........................................................................ 78

A.1 KANSEI USER RESPONSES.............................................. 78

A.1.1 Response 1 ................................................................... 78

A.1.2 Response 2 ................................................................... 82

A.1.3 Response 3 ................................................................... 87

A.1.4 Response 4 ................................................................... 90

A.1.5 Response 5 ................................................................... 95

A.1.6 Response 6 .................................................................. 100

A.2 KANSEI DEVELOPER RESPONSES: .............................. 105

A.2.1 Response 1 ................................................................... 105

A.2.2 Response 2: .................................................................. 108

A.3 JFTP DEVELOPER RESPONSES: .................................... 111

A.3.1 Response 1 ................................................................... 111

A.3.2 Response 2 .................................................................. 114
A.3.3 Response 3 ........................................................................... 117

A.4 JFTP USER RESPONSES: ........................................................ 119

A.4.1 Response 1 ........................................................................... 119

A.4.2 Response 2 ........................................................................... 121

A.4.3 Response 3 ........................................................................... 124
LIST OF FIGURES

Figure 1: General Structure of an OSS Community................................. 3

Figure 2- Geoffrey Moore’s Technology Adoption Model ...................... 5

Figure 3- Annual counts of research publications................................ 7

Figure 4- Constructs studied in the reviewed research papers and their relations................................................................. 11

Figure 5- DeLone and McLean’s Model of IS Success ......................... 21

Figure 6- Updated DeLone and McLean IS success model.................... 25

Figure 7- Findings of the longitudinal study ..................................... 26

Figure 8- Distribution of major file types worked in projects per year .... 38

Figure 9- Research Methods and Level of Analysis ............................ 49

Figure 10- Progress of the case study projects in the sustainability model .................................................................................. 72
CHAPTER 1

INTRODUCTION

In the 28 years from 1970-1998, open source software began and evolved into a movement without assuming a name. Since the term “Open Source Software (OSS)” was coined in 1998 [16], we have been seeing a lot of interest in this topic. Many organizations are adopting open source software and the number of open source projects is increasing faster and faster. The growth of open source software has caused significant changes to the business models of commercial organizations. Dependence of government and of other organizations on OSS is growing. Bollinger, et al. [1] found that open source software plays a far more critical role in the U.S. Department of Defense than has been generally recognized.

Ye, et al. [2] define open source software as “those systems that give users free access to and the right to modify their source code.” Hence, there is a lot more code reuse in the open source products than the proprietary products. The Open Source community attracts very bright, very motivated and disciplined developers. “In addition, these developers are not part of corporate cultures where the best route to large salaries is to move into management; hence some Open Source developers are amongst the most experienced in the industry.” In addition most users
often suggest both bug fixes and enhancements as actual changes to the source code. "Consequently the quality of software produced by the Open Source community sometimes exceeds that produced by purely commercial organizations." [3]

Eric Raymond [4] compares the two fundamentally different development styles of Open source and traditional methods. He says that the traditional development style is represented by the ‘cathedral’ model whereas open source development style is represented by the ‘bazaar’ model. The traditional software development is centralized with roles being clearly defined similar to building a cathedral whereas in the open source model, it is decentralized and the roles are not clearly defined. The open source community of developers consists of people of varying skills and experience. The open source community is very large and is continuously growing. The developer hierarchies of open source projects are flat mostly as the roles are not clearly defined. Nakakoji, et al. [5] state that all the actors of OSS fall in one or more of the roles forming the general structure of an OSS community. The roles specified by them are

1. Project Leader – responsible for the vision and the overall direction of the project.

2. Core Members – responsible for guiding and coordinating the development

3. Active Developers – contribute regularly.
4. Peripheral Developers – occasionally contribute new features.


7. Readers – active users who try to understand the system by reading the code.

8. Passive Users – just use the system.

Figure 1: General Structure of an OSS Community [5]
Sourceforge currently has more than 190,000 open source projects in various development phases. Slowly, some of the open source products are beginning to dominate their commercial counterparts. According to a survey by netcraft [6], as of May 2011, 62.71% of the web server market share belongs to Apache whereas only 18.37% belongs to Microsoft.

Previous work on open source software has focused mainly on two things, namely, the motivations of open source software developers and the success of open source software. While there have been some very successful open source software projects, many of them have failed. As of 2005, among the projects in Sourceforge, most of them have ended in failure: 58% did not move beyond the alpha developmental stage, 22% remain in the planning phase, 17% remain in the pre-alpha phase, and some become inactive. [7]

A major barrier to the adoption of open source software is the sustainability aspect of it. Geoffrey Moore’s technology adoption model classifies the potential user base based on when they adopt. He says that if a software product does not cross the chasm, it will not have sufficient users to be sustained.
Figure 2- Geoffrey Moore’s Technology Adoption Model [8]

[8] “If we apply this model to open source software adoption, then on the left hand side of the chasm is the community source code, documentation, and infrastructure that meet the investment criteria of the core user group – the innovators. On the right hand side of the chasm is the larger products and services market of the later-stage adopters. This larger market requires that a product not only meets user requirements regarding software functionality and quality, but also that it comes with the documentation, support, and implied warranty of continued availability that non-technical and organizational users need to tempt them into trying the product.” An estimate is that as many as 90% of open
source projects struggle to cross the chasm from one model of sustainability to another. [9]

This has led to this research of the success indicators of open source software. As open source software development is very different from its commercial counterparts, many of the success indicators applied to the commercial projects like budget planning, meeting requirements and timely delivery, market share, etc. cannot be easily applied to open source development.

1.1 The Problem:

Although a lot of previous work has been done on identifying the indicators of success, most of the work has been exploratory and case study based, while some of it has been empirical in nature. A survey [10] has captured the growing interest in this topic through 1999 to 2005 as shown in the figure below.
Though empirical research has identified different indicators of success, most research has used only one of them to assess success. Most empirical research projects have also used only the data available from sites like Sourceforge and freshmeat and so they have identified the measures based on the projects they studied. These projects have also not specified a generic framework that a prospective developer or a project administrator can use to assess their projects. There is also not a set of guidelines that one can use for collecting the data required for the assessment and for applying the collected data to assess the success of the project.

Through this thesis we seek to develop a framework of measures for analyzing the success of open source software that one can use to apply to one’s own projects. We also intend to describe the data collection
methodologies in detail that can be utilized to collect the required data and to apply the framework to any open source project. We will also apply the complete framework to two very different case study projects.

1.2 Solution Approach:

An extensive literature survey was done to identify proven measures of success. Some of the measures that can be applied to open source software were also observed in software engineering best practices. Methodologies for collecting data and for applying the framework were established and described and applied to projects chosen as case studies. We will now review the usage of case study research and explain why we chose the case study approach.

Case Study Research [11]:

The factors that affect which research strategy is chosen to answer a particular research question are:

“– The type of research question

– The control an investigator has over actual behavioral events

– The focus on contemporary as opposed to historical phenomena “

“A case study is an empirical inquiry that
• Investigates a contemporary phenomenon within its real-life context, especially when

• The boundaries between phenomenon and context are not clearly evident”

An investigator has very less control over the actual behavioral events when the question is of the type ‘how’ or ‘when’. When there is less control and when the focus is within real-life context, case study research is the most suitable strategy.

Hence we chose the case study approach to demonstrate ‘how’ our framework can be applied to any project.

The rest of the thesis is structured as follows. Chapter 2 is the literature review of related research in this topic and it also capture factors that maybe probable factors which could be added to our framework with additional research. Chapter 3 lists and explains our framework in detail. Chapter 4 explains the data collection methods for our framework and also lists our surveys. Chapter 5 is the analysis of our case study projects by applying our framework. Chapter 6 contains the conclusion and future work.
Since the rise of the open source model, there have been a lot of questions about it from the economic point of view. The question asked is, “How can a group of unpaid developers of varying experience and skill sets get together and create quality software products.” This has prompted research on the motivations of developers. As most of the open source products are failing while some of them are very successful, the other main area of research has been on the success of open source software.

A study\textsuperscript{[10]} of all the current existing literature conducted very recently shows that the sample sizes of projects in open source software research was highly skewed towards single projects (42%). It also shows that the case study approach was the most common, which was 43% of all papers, and these studies mostly performed case studies at the group level and it was based on archival data in more than half of the instances. Studies just based on archival data cannot clearly characterize the satisfaction of users or the satisfaction and motivation of developers, since only inferences can be made from the statistics and the numbers.
The following framework was developed from a study \cite{10} of all the existing literature on open source software (OSS) by identifying the major concepts in the OSS research papers in each of the categories.

Figure 4- Constructs studied in the reviewed research papers and their relations \cite{10}
The study says that the most studied constructs were also strongly reliant on archival data whereas the studies of motivation tended to use questionnaire data more often. The most studied constructs from the study are listed as follows:

1. Project characteristics (21%) which were overwhelmingly studied through archival data

2. Social processes (16%)

3. Success (12%)

We will now review existing work on some of these most commonly studied constructs.

2.1 Motivation for OSS Developers:

Ye, et al. [2], find that new members are attracted to an OSS community because the software can solve one of their problems. The new developers start out as readers from being passive users and then slowly try and fix the bugs they encounter as they get more familiar with the system. “The depth and the richness of good OSS systems often drive motivated members to want to learn more, to read the system” [12]

Lerner, et al., [13] state that a developer participates in an open source or commercial project only if he/she derives some benefits from it. It is generally observed that some members get involved in a project as
core members and keep contributing to it. There are also some organizations and developers who find that their need will be satisfied by a project if its features are slightly modified, or a set of new features is added. So they work on it, to add what they want, contribute it to the community, and then stop working on the project completely. Some new developers view open source projects as an opportunity to get hands-on experience and to improve their skills. In other words, learning has been cited has one of the major motivations. Some immediate benefits that open source developers receive are improved performance in the mission assigned to them by their employers, getting to do enjoyable work. Some delayed benefits are career incentives such as building reputation for employment with prospective employers, and fulfillment of the desire to get recognized by peers.

Ye et al. [13] also state that the existence of numerous OSS projects facilitates the integration of OSS with university classes which gives the students a chance to read quality code and work with talented open source developers, thus leading the way to the next generation of open source developers who will sustain the open source movement. To become good writers of code, the students have to learn to read first and the importance of that skill should be stressed.
2.2 License Types:

Open source licensing is a way of sharing software through the internet with the limitations being decided by the creator. They are able to control their work through the source code, allowing them to make their work Transparent, Recombinant, Credited, and Circulating. [15]

"**Transparent** requires the source code to be transparent, so that people can see how it was made.

**Recombinant** Allows other artists to recreate original piece by editing, rearranging, or adding of new material.

**Credited** Shows who the original creator was and who edited what.

**Circulating** Ensures that the original piece of art and any variations made remain in circulation to the public." [16]

The license of an open source project is very important as it influences the decision of the developers and users to devote time to that particular project. There are many types of licenses used in OSS. The open source initiative [14] lists them. The following are the most popular ones in use currently:

- Apache License, 2.0
- BSD licenses (New and Simplified)
- GNU General Public License (GPL)
• GNU Library or "Lesser" General Public License (LGPL)

• MIT license

• Mozilla Public License 1.1 (MPL)

• Common Development and Distribution License

• Eclipse Public License

Fershtman and Gandal have classified licenses into three classes of relative restrictiveness, namely, very restrictive, moderately restrictive and non-restrictive. [17]

• The GNU general public license or the GPL makes source code available but states that “other programs that incorporate code from a GPL licensed software must also make its source code fully available under the GPL.” This is classified as very restrictive. This is the most well-known license for OSS projects.

• The GNU Lesser GPL or the LGPL is less restrictive than the GPL. It is classified as moderately restrictive and is the most popular in the less restrictive category.

• The most popular among the non-restrictive licenses is the Berkeley Software Distribution or BSD license. This only requires due credit is given to the copyright holders of the code
and thus is less restrictive than the other two categories of licenses.

2.3 Network Characteristics:

While we do not explore the role of social networks of OSS communities in this thesis, we give an introduction to recent research on this topic. Research in this topic is in its early stages and more research is required to confirm the role of network characteristics in the success of an open source project than just archival data.

A dissertation by Jin Wang \[1^{18}\] has identified certain network characteristics to affect open source projects positively based on Sourceforge which are described as follows.

2.3.1 Network Size:

The study indicates that projects with medium to large networks have higher performance measures than those in smaller networks. It also states that both extremely small and extremely large networks limit the conduct and performance of open source software projects.

2.3.2 Network Density:

Analysis shows that Open source project conduct and performance is positively influenced by density of the network in dense networks than in sparse networks. It further adds that “closely knitted networks are conducive for developing trust and enhancing reputation mechanisms
which encourage desired behavior and constrain undesired behavior.”

Though they seem to create a behavior pressure for members to conform rather than innovate. Jing Wu, et al. [18], finds that project density negatively affects the project development activity.

2.3.2 Network Diversity:

The study indicates that network diversity is a positive influence as it provides them with novel and distinct perspectives, information and knowledge that may not be available for them otherwise. Though, diversity seems to come with the expense of additional co-ordination costs.

2.3.3 Project and Developer Degree Centrality:

There is indication of positive impact of project degree centrality which is analogous to the number of developers in a project. There seems to be mixed opinions on this aspect. Stewart, et al. [19] finds that the team size does not affect the completion of tasks. The study [18] indicates that developer degree centrality affects the success variables negatively.

2.4 Software Quality:

Lee, et al. [7] finds that user satisfaction was influenced significantly by software quality and community service quality.

Community service quality is defined as “an individual’s perception about
the reliability, responsiveness, assurance and empathy of the service provided by the OSS development community.”

2.4.1 Software Development practices:

As good software development practices are necessary for better quality of the product, we look into the development practices of open source software.

A literature survey by Kevin Crowston, et al. \[10\] indicates that open source projects do not engage in formal planning. Though, they cite that projects mostly have to-do lists or feature requests that form an agenda and those firms involved in open source projects seem to contribute to the planning of the project. Open source projects do not conduct formal software requirements either. Requirements are mostly found in email messages from discussions among the developers and the users as to what the software should do and should not do. Modularity of code is one of the strong points of the open source model.

Different projects follow different testing procedures. Developers test their own code in most projects, though they are open to inspection by others later. As Eric Raymond said, “Given enough eyeballs, all bugs are shallow” \[4\] . This is the reason that many open source projects have high quality code with very few bugs. Many projects seem to follow his suggestion to “Release early and release often”, though many users seem to prefer releases of known quality with stability.
Maintenance is a very important activity for open source projects. As suggested above, community service quality is very important for the satisfaction of users. Satisfied users lead to the popularity and success of a project. So open source developers responding to bug reports and feature requests of users promptly is very important.

2.5 Team Characteristics:

Some characteristics of the development team have been studied extensively in small groups’ research. We have not considered the interactions between the team in our research due to limitations in the data. We give an introduction to the little research that has been done in this area.

2.5.1 Trust:

Stewart et al. [18] indicate trust as an antecedent to OSS team input effectiveness. “The words, actions, and decisions of others may affect the emotional and psychological consequences that an individual experiences from participation” They cite previous work which indicates that low levels of trust within a team are related to decreases in project participation and contribution among team members and those projects also seem to suffer from high levels of turnover.

Hui-Min et al. [19] found that team members’ trust in the leader of a project affects the performance of the team positively. They also cite McAllister that trust has cognitive and affective foundations. Cognitive
trust is based on reliability and dependability whereas affective trust is based on care and concern for the welfare.

The results of the findings of Stewart et al. find that affective trust affects both team size and team effort positively and cognitive trust does not affect team size and team effort positively. It further adds that collaborative values, individual values and the forking norm had direct efforts on affective trust and that cognitive trust was a significant antecedent to affective trust. The forking norm is the norm that is against splitting a project into two or more similar projects developed separately.

2.5.2 Shared Understanding:

Crowston et al. cite that previous research suggests that shared mental models which guide member actions are important for team effectiveness. They also quote about a fundamental problem in building large systems which seems to be the development of a common understanding of the requirements and design across a team and that the “transcripts of team meetings reveal large amounts of time designers spend trying to develop a shared model of the design.” An analysis by them on four developers from the Apache Lucene project indicate that core members have a high level shared understanding of aspects like key definitions but also note that the sharing is not complete.
2.6 Success Measures:

Previous work has focused mostly on taking the established success measures of Information Systems research and assessing them for their usefulness in the OSS context.

Crowston et al. (2003)\cite{20} identified a range of measures from information systems research and assessed its usefulness in the OSS context. They show that the DeLone and McLean’s Model\cite{33} is the most commonly cited model for IS success.

![Figure 5- DeLone and McLean’s Model of IS Success\cite{33}]
They find that the literature review suggested the following success measures and their indicators:

- “System and Information Quality
  - Code Quality
  - Documentation Quality

- User Satisfaction
  - User ratings
  - Opinions on mailing lists
  - User surveys

- Use
  - Use
  - Number of users
  - Downloads
  - Inclusion in distributions
  - Popularity
  - Package dependencies
  - Reuse of Code

- Individual and organizational impacts
Based on reexamining the OSS process by including data on the development environment as opposed to focus on only the use environment, they suggested the following:

- **“Project Output**
  - Movement to stable status
  - Achieved identified goals
  - Developer satisfaction

- **Process**
  - Number of Developers
  - Level of Activity
  - Time Between Releases
  - Time to close bugs or implement features

- **Outcomes for project members**
  - Individual job opportunities and salary
  - Individual reputation
  - Knowledge creation"
They posted an open ended question in a forum asking about the things developers look at to know when things are going well or not. They find that developers suggested the following as possible measures:

- Recognition
- Influence of project’s process on other OSS and commercial groups.
- Involvement of the users
- Portability.

They also performed an empirical analysis [21] using data from Sourceforge. They suggested that there was room for improvement in the measures and that additional data could be collected to increase the accuracy of the developer counts and listed some other limitations of their research.

Lee et al. [7] also researched the success measures of OSS based on the DeLone and McLean IS success model. They updated the model and presented their model which is shown below:
They state that the above model assumes that service quality, system Quality and Information quality affect use and user satisfaction, and that use and user satisfaction are dependent on each other and that they are antecedents of net benefits. They performed the data analysis using PLS (Partial Least Squares) and tested the measurement model using Cronbach’s α assessment.

Their OSS success model has five determinants:

- **Software Quality** – Qualities of the software like user friendliness, ease of use, useful functionalities etc.

- **Community Service Quality** – Service quality from the OSS development community.

- **User satisfaction** – Satisfaction of the users of the product.
- OSS use – Frequency and the variety of tasks that the product is used for.

- Individual net benefits – Measured in terms of job and decision-making performance.

Subramaniam et al. [22] studied the determinants of OSS project success using a longitudinal study. They collected Sourceforge data for each month over a period of 5 years and have discussed their impacts. Their findings are shown in their figure below:

![Figure 7- Findings of the longitudinal study](image)
They find that some of the success measures and determinants are interrelated.

Crowston et al. [10] state that “an analysis of the previous work on success measures indicate that OSS success is a multidimensional construct, but most empirical research has only used one of these dimensions to assess success.” For example, Ravi Sen [23] explores the determinants of open source project success as measured by project popularity. He finds that the following factors had effects on the popularity:

- Unix-like operating systems had a positive effect.
- Target audience of desktop users had a positive effect.
- Age and Development status had significant positive effect.
- GPL license had a negative effect.

Our framework on the success of open source software focuses on multidimensional constructs and it uses data obtained directly from the users and developers of a project through surveys and interviews and does not depend solely on archival data from websites like Sourceforge, Freshmeat etc. This framework can be used on any open source project to analyze its success and to improve the project. The framework is also applied on case study projects to demonstrate the application procedure.
CHAPTER -3

THE FRAMEWORK

Based on software engineering best practices and proven success measures in the existing literature, we develop a framework of success measures. This framework can be used by project administrator, prospective developers and prospective sponsors. The administrator can use the framework to analyze the project’s success and determine what needs to be remedied to make the project successful. Prospective developers and sponsors can use this framework to analyze the possibility of success of the project to decide whether they can choose to participate in the project or not.

The success measures of the framework are as follows:

1. Domain size and competition
2. Development status
3. Team Structure
4. Usage
5. License

6. Programming Languages used

7. Portability

8. Developer Satisfaction

9. User Satisfaction

10. User Participation

11. Organizational Sponsor

12. Reputation and Recognition

Each success measure of the framework will be explained in detail along with the ideal result for a particular measure.

3.1 Domain Size and Competition:

The benefits of leading a successful open source project are many. This leads some people to start an open source project without proper analysis of the domain and the existing competition. “Building the right product is more important than building the product right.” Some people might start a new project when an addition of just a single feature to an existing project would have solved their need. So complete domain
analysis and evaluation of the current competition is very important for any new project. If there is a successful open source competitor with a large user base, then there has to be a significant edge to the new project in order to draw the users over to it.

Projects that have a larger user base than their competitors are at a significant advantage. The user base can be compared by the number of user accounts or the number of downloads for competing products in the same domain.

“A domain is defined as a collection of current and future (software) applications that share a set of common characteristics and also has a well-defined set of characteristics that accurately, narrowly, and completely describe a family of problems for which computer application solutions are being, and will be sought.” Domain analysis is considered as the key method for realizing software reuse. As many features of the competitor projects are common and as their code is available, reuse of code is absolutely necessary when possible. A large domain with many competitors with not much difference in terms of usability and features is not good as the potential user base is split into many parts.

A small domain with not many competitors, or a large domain and having edge over the other competitors in terms of features are ideal results for this success measure. It is unlikely that a large domain will not have many competitors.
3.2 Development Status:

The development status of a project is an indicator of the amount of work that has been completed and the stability in the later stages of the development. Sourceforge categorizes open source projects using 7 development statuses.

- Planning – Goal of the software is finalized in this phase.
- Pre-Alpha – This phase includes all activities prior to testing like requirements analysis, design, development etc.
- Alpha – Testing begins in this phase.
- Beta – Focuses on reducing impacts to users and this phase generally begins when the features are complete.
- Stable/Production – This phase comes after testing is complete and assumes that there are no undocumented problems.
- Mature – Projects in this phase are stable, well documented and widely used.
- Inactive – Projects that are dropped are listed in this phase.

Users tend to try a project more often if it is in the stable/production stage or if it is in the mature stage. Subramaniam et al. [22] state that OSS in the later stages of development should be preferred by those interested in using, rather than modifying, the software. They
suggest that the increased usage means more feedback and bug reports, which in turn results in more project activity. They also state that “projects in advanced stages signal a commitment on the part of the project administrators to support, guide and advance the project, which in turn could attract new developers.” In general, projects in the later stages of development generate more activity and are expected to attract more developers.

3.3 Team Structure:

Few previous works have looked into the ideal team collaborations and structures. Jing Wu et al. [24] researched on the communication patterns within the development team as a potential determinant of open source success. The indicators they identified were

“(1) Project centrality, which indicates the inequality of the developers’ contributions in the project; and

(2) Project density, which represents the readiness of the group to respond to changes and how close a network is to realize its potential.”

They indicate that high project centrality leads to more development activity as the core members can filter the important information and resources to the group, which improves the efficiency and communication within the team. They also hypothesize that project density affects development activity negatively as there will be repeated exchange of information in large groups, thus taking longer to
communicate effectively throughout the team. They cite studies which state that effective communication between the teams is important for the success of the project.

Mockus et al. [25] study the Apache and Mozilla projects and mention a set of hypotheses based on their findings. Their hypotheses related to the team structure are listed verbatim as follows:

"Hypothesis 1: Open source developments will have a core of developers who control the code base, and will create approximately 80% or more of the new functionality. If this core group uses only informal, ad-hoc means of coordinating their work, the group will be no larger than 10 to 15 people.

Hypothesis 2: If a project is so large that more than 10 to 15 people are required to complete 80% of the code in the desired time frame, then other mechanisms, rather than just informal ad hoc arrangements, will be required in order to coordinate the work. These mechanisms may include one or more of the following: explicit development processes, individual or group code ownership, and mandatory inspections.

Hypothesis 3: In successful open source developments, a group larger by an order of magnitude than the core will repair defects, and a yet larger group (by another order of magnitude) will report problems."
Based on these studies, we state that the ideal success result for the team structure measure is a small team of core developers with a larger team around the core team.

3.4 Usage:

User base of software is a very important measure in assessing the success of software. This can be analyzed for commercial software by their market share. For open source software which does not require users to have accounts, this might be difficult to capture. A good alternative to measure the usage of an open source software is the number of downloads. It does not mean that each download is a user of the software; it means it is a potential user. If the satisfaction of the users (measure #9 in this framework) is high in general, then we can safely assume that almost all potential users are actual users of the system. If the satisfaction level is low, then the actual number of users might be much lower.

Ravi Sen [23] chose the size of installation base as the measure of open source project success. He chose the project popularity of a project on the Freshmeat website to assess the size of installation base. They define project popularity as “function of the number of times information is accessed about this project at Freshmeat’s website, the number of hits (i.e. visits) received by the project’s own website, and the number of subscribers of this project.”
Crowston et al. \cite{20} has the same perspective on the popularity and the number of downloads of a project being a good measure of use. They also add that inclusions in distributions are an important success measure. They also consider the use of a project by many other projects to be an important measure.

The more the number of downloads or user accounts the better as it gives a good indication of greater market share. The number can be compared with the major competitors in the same domain to be more accurate.

3.5 License:

Some previous work has focused on the ideal license for open source software. The most popular licenses are GNU General Purpose License (GPL), GNU Lesser GPL (LGPL), and Berkeley Software Distribution (BSD). There have been mixed results with respect to the ideal license for open source software success.

Ravi Sen \cite{23} found that if a project chooses GPL as the end user license for its software, the popularity of the project goes down by 64%. They believe that this adverse effect is because of the restrictive nature of GPL. A commonly cited example for the restrictive nature of GPL is that if a licensee includes any amount of GPL code in another program, that entire program becomes subject to the terms of GPL. They indicated that the GPL would be the most likely license for projects whose target
audience is desktop users as they do not contribute to the development of the project. Our case study also indicated that the end users who just want to use the software never care about the license of an open source project. They further add that when the intended users are developers or system administrators, the popularity decreases by choosing GPL as they prefer less restrictive licenses in order to reuse code.

Stewart et al. \cite{26} hypothesized that “OSS projects using a restrictive license will attract greater development activity over time than those using a non-restrictive license.” Their argument is that developers take comfort in the fact that a commercial organization cannot create a commercial version of their work which might become dominant so that the developers might end up paying for software that grew out of their efforts and be unable to modify the commercial version to best suit their needs. They were not able to support this hypothesis.

A search on the blogs of many developers and the development community in general, we find that majority of the developers prefer non-restrictive licenses. The main point they seem to make is that they do not want to use GPL code as it forces their code to be under the terms of GPL. However, the majority of the projects on Sourceforge are licensed under GPL. In other words, research appears to indicate that there is still not a clear ideal license choice. More research is required in this topic as there is no clear choice as of now. For the purpose of our framework, we go
with Ravi Sen in the choice of GPL license when the target audience is desktop users and non-restrictive license otherwise.

3.6 Programming Languages Used:

As most open source software projects are heavily dependent on other OSS developers to contribute to their project, using a commonly used language is very important, because this increases the probability of attracting more developers.

Emanuel et al. [27] recorded the statistics of the programming languages used by open source projects in Sourceforge as of February 2010. They found 127,247 projects which listed their programming language and they had about 97 programming languages. The classification is as follows: “Java (20.10%), C++ (16.27%), Other OOP (7.45%), C (14.91%), PHP (13.14%), Other Script-based (18.57%), or Other (9.56%).”

English et al. [29] compared C++ and Java systems of the same size and from the same application domain. Their findings indicate that C++ systems are more difficult to understand and maintain than Java systems.

Karus et al. [28] used data of projects from ohloh.net which tracks about 400,000 open source software projects. They analyzed 22 projects from 1997 to 2009 and plotted the distribution of major file types worked on in projects shown in figure-8. They found that the number of C/C++ files showed a steep decrease whereas that of Java showed a sudden and
strong what yet no longer growing presence. They conclude that the most popular language used in OSS software is XML followed by Java and C. They observe that “XML has increased its popularity steadily over the last decade while C has lost its high share to several other languages of which Java has been among the most popular ones.” The ideal result for our measure should be one of the popular languages mentioned above.

Figure 8- Distribution of major file types worked in projects per year [28]
3.7 Portability:

There are advantages and disadvantages of software being compatible with only one particular operating system. Subramaniam et al. \cite{22} hypothesized that “OSS projects developed to run on UNIX or Linux, rather than other operating systems, will attract more developers and lead to greater project activity.” They state that the use of UNIX/Linux operating system in an open source software project is associated with its success. They find that the use of the UNIX/Linux operating system does indeed lead to higher development activity. They also find the positive impact of OSS projects built on Windows operating systems in all their project success measures. They also find that projects using UNIX/Linux have a negative impact on user interest as not many non-developer users use UNIX/Linux operating systems.

Ravi Sen \cite{23} found that projects which develop software that run on Unix-like operating systems are more popular and that the choice of other operating systems did not have any significant impact on project popularity.

Software that is cross-platform seems to have the advantages of all the different operating systems. The ideal result for the portability measure is that the software should be cross-platform. Crowston at al. \cite{20} found that developers consider porting to different systems especially
windows and requests for such ports as a measure of success of the product.

3.8 Developer Satisfaction:

Crowston et al. [20] cite that “goals for an OSS will most likely come from a discursive process centered on developers.” So they suggest that a key success measure for OSS might just be developer satisfaction with the project. This can be measured by contacting the developers. Their analysis indicates that the developers find their involvement, satisfaction and enjoyment as measures of success of a project. The authors find this opinion consistent with the view of OSS as “software that scratches an itch.” If a developer is completely satisfied with the project he/she is working in, it means that the software has either succeeded or is well on its way to success.

Wu et al. [30] proved the following hypotheses’ based on their analysis on the motivations of developers motivations:

- “Developers’ satisfaction with prior participation in OSS projects will have a positive effect on their intentions to contribute to future OSS development.

- Developers’ motivation to help will have a positive effect on their satisfaction with participating in a project.
• Developers’ motivation associated with career advancement will have a positive effect on their satisfaction with participating in a project.”

According to their results, satisfaction with participating in OSS projects had the strongest influence on OSS participants’ intentions to participate in future projects, followed by their motivation on enhancing human capital and satisfying personal needs. Thus satisfied developers are very important for any OSS project.

3.9 User Satisfaction:

User Satisfaction is a very important measure of success that is often mentioned by previous work in this area. Satisfied users tend to contribute to the project as developers or as bug reporters. They also spread the word to people with similar needs which in turn leads to more usage. Crowston et al. [20] mentioned user ratings on projects as a measure of user satisfaction, but they also hypothesized that users who do not like a software product, did not bother to rate it poorly as they found scores that ranged from 7.47 to 9.07 in a sample of 59 projects. They also mentioned that opinions expressed on project mailing lists are a potential source of qualitative data. They suggested that surveys could be built in the software and cited the Mozilla web browser including a program for collecting crash reports and other feedback as a good example.
Lee et al. [7] hypothesized two things with relation to user satisfaction. They were:

- “User Satisfaction has a positive effect on OSS use
- User Satisfaction has a positive effect on individual net benefits.”

They cite previous empirical analysis which tests the relationship between satisfaction and system usage and they find that users who feel satisfied continue using the system. DeLone and McLean explained that net benefits are measured in terms of job and decision-making performance. They argue that if a user is satisfied with an information system, this means that the information system has had an impact on the user’s performance. Their findings indicate that usage of OSS is determined by user satisfaction and software quality and that user satisfaction is influenced by software quality and community service quality.

So, high satisfaction levels of users are a great indication of the success of a software product.

3.10 User Participation:

Involvement and participation from the users are very important for the success of any open source software product. As many open source projects do not have formal testing procedures, reporting of bugs by users
is also very important. This is reinforced by the quote of Raymond who said “given enough eyeballs, all bugs are shallow.” Feature requests by users allow the developers to prioritize the features to be implemented. An ordinary software product can become a very successful project if users participated actively and if developers responded to it promptly.

Crowston et al. [20] analyzed the responses from developers and found that the level of involvement of users emerged as a new theme. They had only considered contributions from developers, but this finding reinforced the fact that “the Open Source projects are dependent on help from users to identify problems and post suggestions.” Here, the level of involvement of users was as indicated by the involvement of users in submitting bug reports and participating in the project mailing lists.

Stewart et al. [26] also cite that the audience for individual contributions will be wider and hence the efforts of contributors will be more visible if the user interest in a project is greater. They hypothesize that the “user interest will have a positive effect on the amount of project development activity over time.” Their analysis is in support of this hypothesis. Subramaniam et al. [22] add to this finding based on a longitudinal study. They suggest that “while change in user-interest has positive impact on project activity levels over short term, lack of user-interest adversely impacts project activity levels over longer periods.”
Some reasons that they suggest for this are [22]

1. Core developers deciding that that software does not need any more improvements and stop paying attention to feedback and requests.

2. Not enough developers to respond to all the requests and feedback.

3. More reports of bugs or feature requests may lead to staggered file releases to accommodate the requests.

4. The project has been abandoned due to financial or other reasons.

More research is required to confirm this finding as we feel systematic and well supported projects with good resources might not face this problem. We suggest that greater levels of user participation indicated that the project has been a success.

3.11 Organizational Sponsor:

Organizational sponsors for open source projects are one of the less researched areas compared to other areas in existing literature. The participation of organizations in open source software has left many wondering the purpose behind it. Bonaccorsi et al. [31] noted the emerging hybrid business models because of the entry of companies producing free software and shifting the value from licensing agreements to other services like maintenance, consultancy, training etc. They also observe that some companies use their own licenses which protect copyright, but release the source code which certain category of users can use. Examples
of such licenses are the Apple public source license and the Sun community source license. Another advantage noted by them is that companies provide monetary incentives for doing the non-interesting work like documentation etc. Co-existence of commercial and open source platforms is a key for the success of the open source movement.

Crowston et al. [10] cite previous work in which surveys have shown that almost 45% of the contributors are paid by firms for their participation directly or indirectly. Firms are motivated to be involved in open source because it allows smaller firms to innovate and as the quality of the software will be high. They also note that firms focus less on social motivations like reputation and learning benefits.

Using data gathered from freshmeat and project websites, Stewart et al. support the following hypotheses': [26]

1. “Projects with a sponsor will attract greater user interest over time than those without a sponsor.

2. The effects of sponsorship on OSS project user interest over time will be stronger for projects that have a broader range of potential uses than for those that have a narrower range of potential uses.

3. Projects with a nonmarket sponsor will attract greater user interest over time than those with a market sponsor.
4. Projects with a nonmarket sponsor and a nonrestrictive license will attract greater user interest over time than any other combination of license restrictiveness and sponsorship.

5. OSS projects that have a nonmarket sponsor will attract greater development activity over time than those that do not have a sponsor.”

The second hypothesis was supported partially and the others in this list were fully supported by them. Thus having a sponsor helps the success of open source software and it is most beneficial if the sponsor is a non-market sponsor like a university.

3.12 Reputation and Recognition:

Reputation and recognition has been cited as one of the most popular motivations for a developer to participate in an open source software project. The developers of successful projects make useful contacts and it looks good on their resume which leads to their career growth and they also sell commercial license to organizations sometimes. Developers also participate because of the public recognition they get for their skill and knowledge.

Ke et al. \cite{32} observe many motivations for developers to participate in OSS from existing literature. We have listed only those motivations from their list which are related to reputation and recognition as they
seem to be the most popular motivations from our survey of the existing literature:

1. To make profits by selling the software in future (Lakhani and Wolf 2005)

2. For gaining financial benefits (Hertel et al. 2003)

3. For improving future job prospects (Bates et al. 2002; Ghosh et al. 2002; Hars and Ou 2001)

4. For signaling one’s capabilities to potential employers (Lerner and Tirole 2004)

5. To gain recognition from peers (Ghosh et al. 2002; Roberts et al. 2006; Lakhani and Wolf 2005)

6. To show off one’s work (Krishnamurthy 2006)

7. For enhancing one’s reputation in the community (Roberts et al. 2006)

Crowston et al. state that they found many developers suggesting recognition as a measure of project success. Lerner et al. suggest that the signaling incentive motivates the developers to participate and cites that this incentive is stronger,

1. The more visible the performance to the relevant audience (peers, labor market, venture capital community)
2. The higher the impact of effort on performance, and

3. The more informative the performance about talent.

Thus we state that if the developers of a project get reputation and recognition from the community and the labor market, the project is successful.
CHAPTER 4

DATA COLLECTION METHODS

To apply the framework that was developed in the previous chapter to any open source software project, we have to collect the required data. Crowston et al. [10] selected the related papers in the existing literature of open source software and tabulated the research methods and the level of analysis that had been used by those papers. The table is shown below:

<table>
<thead>
<tr>
<th>Research Methods</th>
<th>Levels of Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
</tr>
<tr>
<td>Total</td>
<td>4%</td>
</tr>
<tr>
<td>Case Study</td>
<td>43%</td>
</tr>
<tr>
<td>Survey</td>
<td>25%</td>
</tr>
<tr>
<td>Objects¹</td>
<td>10%</td>
</tr>
<tr>
<td>Field Study</td>
<td>9%</td>
</tr>
<tr>
<td>Secondary data²</td>
<td>4%</td>
</tr>
<tr>
<td>Instrument Development³</td>
<td>4%</td>
</tr>
<tr>
<td>Multi Method</td>
<td>4%</td>
</tr>
<tr>
<td>Interview⁴</td>
<td>4%</td>
</tr>
<tr>
<td>Simulation</td>
<td>2%</td>
</tr>
<tr>
<td>Experiment</td>
<td>1%</td>
</tr>
</tbody>
</table>

Figure 9- Research Methods and Level of Analysis [10]
From the table, we observe that the case study approach has been the most common choice, but they add that case studies were typically based on group level of analysis and were based on archival data in more than half of the instances and surveys were the next common method.

Here, we choose the case study approach to show how to apply our framework to any open source project. We use a combination of the surveys and the interview methodologies to collect our data. Interviews were used only after the surveys to collect data which was not covered in the surveys. Some of the interviews were through email and some were in person or telephonic.

Surveys should have questions that collect the required data with the framework as a guideline for forming the questions. One can contact users of one’s project through the messaging feature of Sourceforge. The Sourceforge usernames can be found through the forum of the project or through feature requests and bug reports. We applied the framework to two case study projects and did a preliminary analysis of one new project too. We list the surveys we used at the end of this chapter as an example and any data required by the framework which was not collected through the surveys, were done through interviews.
4.1 Case Study Projects:

4.1.1 JFtp:

JFtp is a graphical network and file transfer client. The website of JFtp describes its features as follows. “It supports FTP using its own FTP API and various other protocols like SMB, SFTP, NFS, HTTP, and file I/O using third party APIs, includes many advanced features such as recursive directory up/download, browsing FTP servers while transferring files, FTP resuming and queuing, browsing the LAN for Windows shares, and more. You can also have more than one connection open at a time in a Mozilla-style tabbed browsing environment. The FTP API is separated from the GUI and can also be used in third-party applications.” This project is listed in Sourceforge and is popular.

4.1.2 KANSEI:

KANSEI’s website describes the project as follows: “Kansei is a testbed of 210 Extreme Scale Motes (XSM) hooked individually onto 210 Extreme Scale Stargates (XSS). The stargates are connected using both wired and wireless ethernet. Kansei provides a testbed infrastructure to conduct experiments with 802.11b networking and XSMs. It was initially conceived to test the middleware services for the Tier-2 network of the Extreme Scale (ExScal) project funded by DARPA. The Kansei project is headed by Anish Arora. Kansei exports a web interface on which experiments can be scheduled and the results retrieved.”
Simply put, it is a network sensor test bed of the Ohio State University which exports a web interface where users with accounts can schedule experiments and obtain results.

4.1.3 On-time Measure:

The project website describes the project as follows:

“OnTimeMeasure-GENI is a measurement service for the GENI facility users. The measurement service can be used to perform centralized and distributed orchestration and provisioning of measurements within experiment slices for purposes such as:

- Network paths monitoring
- Network weather forecasting
- Network performance anomaly detection
- Network-bottleneck fault-location diagnosis”

This is comparatively a new project and is a very interesting case study.

We perform a preliminary analysis on OnTimeMeasure and we apply the complete framework to the KANSEI and JFtp projects.

The surveys used for the KANSEI and JFtp projects are listed as follows.
4.2 **Surveys:**

4.2.1 **KANSEI User Survey:**

1. What are you using KANSEI for?

2. How long and how many and what kind of experiments have you used KANSEI for in the past?

3. Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

4. Is there any period you use KANSEI more often (for e.g.: conference deadlines etc.)? If yes, is there any reason for doing so?

5. Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?

6. Have you encountered any problems with installing or while using KANSEI? If yes, please elaborate.

7. Was using KANSEI for the first time easy? If not, what would you suggest to help with this?

8. Have you setup your own network bed and installed KANSEI on it? (If no, please skip questions 9-12)

9. Are you satisfied with your installation of KANSEI? If no, Please elaborate
10. Did you require assistance to install KANSEI on your own network bed? If yes, what kind?

11. What do you suggest we can do to help users in the future who want their own installations?

12. How long did it take for installing and setting up KANSEI?

13. Is KANSEI easily portable to a different architecture?

14. Have you ever looked at the source-code of KANSEI? If yes, Why? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

15. Is the code well documented and easy to understand?

16. Have you ever needed customizations of KANSEI? If yes, please elaborate.

17. Do you feel you can perform the necessary customizations in the future with your own group?

18. Your comments on the quality of the documentation and user interface? Does it need improvements?

19. Your comments on the reliability, availability, scalability, interoperability and performance of KANSEI?

20. How did you hear about KANSEI?
21. Will you recommend KANSEI to others in the future? Why?

22. Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?

23. Any general comments you have with your experience of KANSEI?
4.2.2 KANSEI Developer Survey:

1. What is the approximate number of Users of KANSEI?

2. Are there any equivalent competing open-source or commercial products?

3. Do users need training for using Ohio State’s setup of KANSEI?

4. What is the approximate percentage of users who install and use KANSEI on their own hardware?

5. How much guidance or training do users need for adopting and using KANSEI on their own hardware?

6. What percentage of the accounts are unused?

7. What percentage of the accounts is used just once?

8. How many users can use Ohio State’s setup of KANSEI at the same time?

9. Is active scheduling needed to avoid overlap of users request to use KANSEI in general or during an active period (near a conference deadline)?

10. Are there any bug reports usually?

11. Is there an active development team which looks after the next updates of KANSEI or do they just maintain it (correcting a bug when reported etc.)?
12. What is the approximate cost incurred for maintaining KANSEI?

13. What percentages of the users recognize or cite KANSEI in their papers?

14. How much will it cost approximately to set up a small network bed and to install KANSEI on it?

15. What percentage of the users request customizations?

16. Are there any users who have worked on the code of KANSEI themselves for customization?
4.2.3 Jftp Developer Survey:

1. How did this effort begin? Who were the first set of stakeholders?

2. Would you consider Jftp as a success? On what criteria do you base this conclusion?

3. What started the adoption of Jftp in the community? How did people start adopting it?

4. How is this project sustained? Is it funded? If yes, how?

5. What do you believe are the different motivations for the multiple developers on this project?

6. In what areas does Jftp need to improve? Give examples, specifically about the UI.

7. As there are multiple open source ftp clients available, what is the unique contribution of Jftp? What was the primary need that was met by Jftp?

8. What is the collaboration/decision process for Jftp?

9. How did the initial development team of Jftp form? How many developers were present initially when the project was started and how many developers were there towards the end of major development?
10. Was the major chunk of development done by the initial team which started Jftp or was there a lot of contributions by the developers who adopted Jftp later?

11. Is there anything you would like to let us know about the factors for the success and the sustainability of Jftp which we have not asked above?
4.2.4 Jftp User Survey:

1. Are you satisfied with Jftp? In what areas can it improve?

2. As there are many other FTP clients, what was the reason behind choosing to use Jftp?

3. What are the unique features of Jftp when compared to other open source ftp clients?

4. Do you know what license is used by Jftp? Does the type of license influence your choice of open source software? Please explain.

5. Have you ever requested features or reported bugs? How promptly does the development team respond to requests or bugs in general?

6. What do you think of the quality of the code and documentation of Jftp? Does it need improvements?

7. How did you hear about Jftp?

8. Would you recommend Jftp to others with similar needs?

9. Is Jftp one of the best FTP clients you have ever used or do you prefer any other FTP client better? If so, which one?

10. How long have you used Jftp? Will you continue using it in the future?

11. Have you ever contributed code to Jftp or other open source software? If yes, what factors do you consider before deciding to work on open
source software? (Few examples are programming language used, high scope for success, personal use, active development team, license, sponsor etc.)
CHAPTER – 5

ANALYSIS OF THE CASE STUDIES AND ITS RESULTS

In this chapter, we will list each measure of the framework and describe the value of that measure for KANSEI and JFtp. We will then discuss the results of the framework and discuss about OnTimeMeasure briefly.

5.1 Application of the Framework:

1. Domain size and competition:

The domain size of KANSEI is rather small as wireless sensor test beds are mostly used for research experiments mostly by universities. KANSEI’s web interface is used by users to schedule experiments on Ohio State’s test bed. Many universities have their own test bed and software to manage it, but KANSEI is one of the few open initiatives in this domain. So universities without test beds can utilize KANSEI for their experiments. Though the domain is small and with many universities having their own beds and software, the scope is comparatively less, but it attracts some researchers from other universities as it is one of the few open and free software to run their experiments.
The domain size of JFtp is very large. File transfer clients are used by a variety of people in all domains, but that fact also brings a lot of competition along with it. Though, if someone is looking for a cross-platform, stable, ftp client, there are only 10 such choices in Sourceforge out of 1564 ftp projects and Jftp is one of them. JFtp has the added advantage that it supports a variety of protocols like FTP, SMB, SFTP, NFS, and HTTP.

2. Development status:

The development status of KANSEI is stable. There are quite a number of things that need to be improved to attract more users and to split the user base.

The development status is Production/Stable. Active development seems to have stopped taking place. The development activity is low but the software seems to be stable with not many bugs. Though there are still few open feature requests.

3. Team Structure:

The KANSEI team mainly consists of computer science graduate students headed by Professor Dr. Anish Arora at the Ohio State University. It is a small core team with few developers and all the development activity including bug fixes are done by them.
The JFtp project was started by one developer and he contributed the major portions and there was a maximum of three concurrent core developers at any point in time. Many developers offered bug fixes and some had features ready. These developers were around the small core team in the team structure hierarchy.

4. Usage:

User accounts are necessary for using KANSEI. One account is offered to a PI of a group interested in using KANSEI and the students under the PI can use the same account. So it likely that the same account will be used by more than one person. Though, based on our analysis, one account is used by one or few users. KANSEI has about 20 user accounts and it has also been adopted and ported by Wayne State University to their own test bed and they have a small development team which takes care of development activity at their end. It’s also been ported by Oklahoma State University, though for testing purposes only.

The download activity for Jftp started around March of 2001 in Sourceforge and is still active. The number of downloads as of today, May 21, 2011, is 117,756 and keeps increasing by the day. The majority of downloads, around 78%, is for windows platform followed by Linux and Macintosh and the top country in terms of downloads is China with a 20% share.
5. License:

KANSEI is planning to be licensed under GPL. This goes with the notion that GPL is the most popular license.

JFtp is licensed under GPL, but it gives the option of commercial licenses for people who wanted to use the source code in closed source projects. They further add that a commercial license includes email, bug-fix support and also a stripped down API package if needed. The price of the commercial license varies according to the importance of the JFtp code in the project and the size of the project.

6. Programming Languages used:

KANSEI uses a variety of language as it involves both embedded and web programming. The experiments are scheduled through a website and they are run on the motes at the test bed and the results are returned to the users. The programming languages that are used by KANSEI are PHP, C, Perl and python.

The main programming language used by Jftp is java. However, it involves XML files as well. The code of JFtp is well designed with separate classes for each protocol which extend from common base classes.
7. Portability:

KANSEI is portable, but the architecture of the test bed needs to be similar if the porting is to be done easily. KANSEI has 3-tier architecture, server-stargates-motes. If the architecture is drastically different, then porting may be possible and would be difficult or it could be incompatible.

JFtp is developed using java, so it is class-platform and can be run on all the different operating systems. Though, the majority (78%) of the users use it on a Windows platform.

8. Developer Satisfaction:

We got a mixed response from the KANSEI developers. They were satisfied with it, but suggested additional features that needed to be added to KANSEI. One developer also added that the interface and documentation looked a little dull and that they were looking to improve the performance speed of KANSEI as it is a bit slow now. However, the developers are actively working on making it better.

The developers of JFtp were satisfied with their project. They felt that the UI design could be improved, but they did not feel that much change was required.
9. User Satisfaction:

Many users of KANSEI were satisfied and some were not satisfied completely. Every user had certain suggestions for area of improvements. Some of them are better log generation, ability to change passwords through the website. There was a bug report and also suggestions for feature additions.

The Users of Jftp seemed to be satisfied and the main suggestion that many people said was the addition of the drag and drop feature. The advantage of JFtp is that it does not require users to install the software and it works reliably on different platforms.

10. User Participation:

The user participation in KANSEI was average. Feedback and suggestions were obtained from some of the users on the request of the developers. There were some feature requests and areas of improvements were also suggested.

The user participation in JFtp was very good. There were bug reports and active feature requests, but the developers are not actively developing on JFtp. It is stable enough and has good features, but there has also been some open feature requests for a very long time. The feature requests and bugs were actively addressed till Jftp became stable and good enough. Though, it does not seem to be affecting the number of downloads and usage of JFtp.
11. Organizational Sponsor:

KANSEI has the Ohio State University brand which is a non-market organization.

JFtp does not have any organizational sponsors.

12. Reputation and Recognition:

KANSEI is well recognized in the Global Environment for Network Innovations (GENI) which is a project supported by the National Science Foundation. It is also cited by the researchers who have used KANSEI in their papers. As wireless sensor network beds and motes are part of a specialized domain, the scope of the benefits that the developers can gain by participating in this project is slightly less, but the developers working on this project are research associates of the university.

We find that the reputation and recognition obtained by the developers of JFtp is much greater. Some of the main reasons is its popularity and that it is developed using Java which is used by most organizations. The lead developer of Jftp has sold some commercial licenses, made some useful contacts and said that it looks great on his resume. The lead developer also added that he can offer small amounts of money for whoever is willing to work on this project even now.
5.2 Discussion of the Analysis Results of KANSEI and JFtp:

Based on the analysis, we find that KANSEI is moderately successful while JFtp is very successful. We explain the reasons for this conclusion as follows.

The domain size and the edge over the competition that JFtp has, has helped it gain more popularity and usage, whereas KANSEI has to gain more popularity to attract more users. KANSEI seems to have quite a number of areas of improvements on which the developers are working actively. The documentation of KANSEI also needs to be improved as suggested by its users. All these improvements will tend to attract more users whereas JFtp has attained a lot of usage over the years and there are very few suggestions that need to be implemented. Both projects have similar core team structure, but the advantage of JFtp is that it had lot more people contributing towards bug reports and fixes due to the size of its user base. We find that the usage of KANSEI is limited and is not widespread, which needs to be improved. The developers of both projects preferred the GPL license as we expected. The main advantage of JFtp is that it used JAVA and is easily portable. This factor was mentioned by most of the Jftp community. The developers and users of JFtp were more satisfied than their counterparts in the KANSEI project and also JFtp had more user participation and the personal benefits of the JFtp developers seemed to be more. KANSEI having the Ohio State brand seemed to help
in gaining the project credibility and reliability and is a helpful factor whereas JFtp does not have any sponsors.

Based on all these factors, we conclude that KANSEI has good scope for success and is progressing towards it as long as the project is improved continuously and the concerns are addressed whereas JFtp is successful and could reach greater heights if there was more active development addressing the feature requests of the users.

5.3 Discussion of the preliminary analysis of On-time Measure:

On-time Measure project is a new open initiative developed by the Ohio State Supercomputer Center and is headed by Dr. Prasad Calyam. It was widely recognized by the GENI consortium and it was one of the 3 suggested candidates for our study by the committee. It uses a variety of languages like Perl, Python, Ruby, PHP and it consists of a small development team which does all the development activity. The reasons, based on an interview with the GENI committee, for the high probability of success of this project is that they have an active development team which is very efficient and they address user feedback and implement their suggestions very quickly and it also has an organizational sponsor. They are beginning to attract more and more users. They also have many packages where the code is reused and the design of the project is also good and there is sufficient documentation which is good. We feel that this new project is well on its way to success.
CONCLUSIONS AND FUTURE WORK

This thesis creates a framework that a project member or sponsor or their prospective counterparts can use to analyze whether the project is successful or if the project has a good scope for success or not. We used a combination of survey and interview based methodologies for collecting the data. We also explained the methods used for collecting the data required for performing the analysis. We applied the framework to two case study projects, namely, KANSEI and JFtp and also performed a preliminary analysis of On-time Measure. We find that JFtp is successful and KANSEI is not very successful, but has a good scope for success as long as the developers address the concerns and improve the project. We also observed that the On-time Measure project has all the signs of a successful project and has a good chance of being successful. We show the position of these three projects in the sustainability model below:
Figure 10- Progress of the case study projects in the sustainability model

Related future work is listed as follows:

- The framework needs to be applied to a larger population of projects and the results should be verified.

- More sophisticated measures based on in-depth interviews with the users and developers can be added to the framework.

- More research on the best license type for OSS is required based on interviews with developers.

- More empirical research on the success measures of OSS is required.

- The Analytic Hierarchy Process (AHP) technique can be applied to the measures of the framework to prioritize them based on the importance of the factor to the success of the project and that will allow project members to prioritize the work that needs to be done to improve the project.
REFERENCES


15. http://three.org/ippolito/writing/wri_online_why.html


A.1 KANSEI USER RESPONSES

A.1.1 Response 1:

Question 1*

What are you using KANSEI for?

Not currently using.

Question 2

How long and how many and what kind of experiments have you used KANSEI for in the past?

Ran CTP. It is a network protocol evaluation experiment. I used it for about two weeks.

Question 3

Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

No plans for future use.
Question 5

Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?

The log generation could be nicer. Many testbeds provide logs collected from a network in a much better way. E.g., nodeid.txt in a single directory.

Question 7

Was using KANSEI for the first time easy? If not, what would you suggest to help with this?

I needed help from one of the students there. Better documentation.

Question 8

Have you setup your own network bed and installed KANSEI on it? (if no, please skip questions 9-12)

No.

Question 13

Is KANSEI easily portable to a different architecture?

I don't know.

Question 14
Have you ever looked at the source-code of KANSEI? If yes, Why? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

Haven't looked at the code.

Question 15

Is the code well documented and easy to understand?

No idea.

Question 16

Have you ever needed customizations of KANSEI? If yes, please elaborate.

No.

Question 17

Do you feel you can perform the necessary customizations in the future with your own group?

No idea.

Question 18

Your comments on the quality of the documentation and user interface? Does it need improvements?
There is some documentation. More documentation is needed regarding collecting the results.

Question 19

Your comments on the reliability, availability, scalability, interoperability and performance of KANSEI?

The testbed seemed reliable. Did not examine the rest of the features.

Question 20

How did you hear about KANSEI?

Papers.

Question 21

Will you recommend KANSEI to others in the future? Why?

Sure. It is a very useful infrastructure.

Question 22

Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?

There are many testbeds and testbed infrastructures. I have used many of them. So, what to use depends on what my goals are at the moment.

Question 23
Any general comments you have with your experience of KANSEI? 

Thanks for maintaining the testbed. The student was very helpful.

A.1.2 Response 2:

Question 1*

What are you using KANSEI for?

I have not used yet, even if I am register. The goal is to run some experiments with a novel transport protocol for research purposes for the academia.

Question 2

How long and how many and what kind of experiments have you used KANSEI for in the past?

No experiments so far.

Question 3

Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

The usage will for sure increase in the following months.
Question 4

Is there any period you use KANSEI more often (for eg: conference deadlines etc.)? If yes, is there any reason for doing so?

We will use it more frequently before submitting papers, that is for sure. The reason is that I have the feeling that experiments are reporting you the best results right when you are arriving at the deadline.

Question 5

Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?

So far, I did not run experiments but looks quite nice plataform. I am just missing a feature to be able to change the password online, without having to contact the system admin.

Question 6

Have you encountered any problems with installing or while using KANSEI? If yes, please elaborate.

So far no problems, as I did not run experiments.

Question 7

Was using KANSEI for the first time easy? If not, what would you suggest to help with this?
I did not use it so far, but it seems to be not too complicated.

Question 8

Have you setup your own network bed and installed KANSEI on it? (if no, please skip questions 9-12)

No.

Question 13

Is KANSEI easily portable to a different architecture?

I don't really know as I did not explored special features of KANSEI like this.

Question 14

Have you ever looked at the source-code of KANSEI? If yes, Why? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

No.

Question 15

Is the code well documented and easy to understand?

I did not look at the code.
Question 16

Have you ever needed customizations of KANSEI? If yes, please elaborate.

No so far.

Question 17

Do you feel you can perform the necessary customizations in the future with your own group?

I did not explore that possibility, therefore I cannot answer.

Question 18

Your comments on the quality of the documentation and user interface? Does it need improvements?

There should be a basic documentation explaining how to run experiments step by step. Sometimes very simple things can become a nightmare when working with testbeds. I did not see such a detailed documentation. Also, it could be part of the GUI an option to change your password or contact email, or add email for more researchers so that a group of researchers can access with their same account to the experiment they are running. I do not mean in real time to modify the parameters of the experience, but just to do basic tasks as downloading results, changing their emails of contact, etc.
Question 19

Your comments on the reliability, availability, scalability, interoperability and performance of KANSEI?

I cannot know as I did not run experiments yet.

Question 20

How did you hear about KANSEI?

Through GENI website, by having a look at which kind of testbeds were available.

Question 21

Will you recommend KANSEI to others in the future? Why?

I guess yes, as far as my experiments work fine.

Question 22

Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?

There are a few inside GENI, I guess you know them. Interesting, in Sweden a testbed for WSN is being developed. The point of this testbed is that the testbed has robots moving the sensors, so you can simulate a MANET with repetible experiments inside.
Question 23

Any general comments you have with your experience of KANSEI?

No extra comments so far.

A.1.3 Response 3:

Question 1*

What are you using KANSEI for?

Measurements of WSNs routing protocols

Question 2

How long and how many and what kind of experiments have you used KANSEI for in the past?

15 ~ 30 mins each job ~10 jobs on a WSN realtime routing protocol

Question 3

Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

Maybe. Decreasing
Question 4

Is there any period you use KANSEI more often (for eg: conference deadlines etc.)? If yes, is there any reason for doing so?

Yes, usually before deadlines Difficulty of getting results out takes longer time than expected

Question 5

Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?

No totally satisfied Sometimes it's not giving back any data even you rerun the same jobs which previously yield data

Question 6

Have you encountered any problems with installing or while using KANSEI? If yes, please elaborate.

support link is broken among other links http://kansei.cse.ohio-state.edu/KanseiGenie/help-support.php

Question 7

Was using KANSEI for the first time easy ? If not, what would you suggest to help with this ?

No. should make node selection easier
Question 8

Have you setup your own network bed and installed KANSEI on it? (if no, please skip questions 9-12)

no

Question 14

Have you ever looked at the source-code of KANSEI? If yes, Why ? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

no

Question 20

How did you hear about KANSEI?

My advisor, Dr. Hongwei Zhang

Question 21

Will you recommend KANSEI to others in the future? Why ?

No. And Yes. I would recommend our NetEye, which is federated with KANSEI

Question 22

Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?
Motelab

Question 23

Any general comments you have with your experience of KANSEI?

again, make node selection easier

A.1.4 Response 4:

Question 1*

What are you using KANSEI for?


Question 2

How long and how many and what kind of experiments have you used KANSEI for in the past?

1 year and more than 20 times

Question 3

Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

Absolutely

Question 4
Is there any period you use KANSEI more often (for eg: conference deadlines etc.)? If yes, is there any reason for doing so?

Yes. It usually happens before Conference Deadlines. Reason: more rounds of test need to be done.

Question 5

Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?

Satisfied.

Question 6

Have you encountered any problems with installing or while using KANSEI? If yes, please elaborate.

Yes. Sometime the webportal is unresponsive and unaccessible.

Question 7

Was using KANSEI for the first time easy? If not, what would you suggest to help with this?

Suggest to have a training/tutorial course on using Kansei.

Question 8

Have you setup your own network bed and installed KANSEI on it? (if no, please skip questions 9-12)
Yes. We run KanseiGenie in Neteye at Wayne State University.

Question 9

Are you satisfied with your installation of KANSEI? If no, Please elaborate

Yes.

Question 10

Did you require assistance to install KANSEI on your own network bed?
If yes, What kind?

No, because I am in the KanseiGenie team:-)

Question 11

What do you suggest we can do to help users in the future who want their own installations?

--

Question 12

How long did it take for installing and setting up KANSEI?

1 month

Question 13

Is KANSEI easily portable to a different architecture?
good

Question 14

Have you ever looked at the source-code of KANSEI? If yes, Why? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

Yes

Question 15

Is the code well documented and easy to understand?

Some of them are documented. But code is good to understand.

Question 16

Have you ever needed customizations of KANSEI? If yes, please elaborate.

Yes. KanseiGenie in Neteye Testbed

Question 17

Do you feel you can perform the necessary customizations in the future with your own group?

yes
Question 18

Your comments on the quality of the documentation and user interface? Does it need improvements?

good.

Question 19

Your comments on the reliability, availability, scalability, interoperability and performance of KANSEI?

At present, it's good.

Question 20

How did you hear about KANSEI?

Team work

Question 21

Will you recommend KANSEI to others in the future? Why?

Definitely

Question 22

Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?

NO
Question 23

Any general comments you have with your experience of KANSEI?

NO

A.1.5 Response 5:

Question 1*

What are you using KANSEI for?

Run sensor network experiments

Question 2

How long and how many and what kind of experiments have you used KANSEI for in the past?

3 years, hundreds of times

Question 3

Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

yes
Question 4

Is there any period you use KANSEI more often (for eg: conference deadlines etc.)? If yes, is there any reason for doing so?

Two periods: before submission deadlines or kansei related proejcts

Question 5

Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?

I am satisfied with its features so far. A more reliable Kansei will be ideal. Also, being able to stop running experiments will be another plus.

Question 6

Have you encountered any problems with installing or while using KANSEI? If yes, please elaborate.

no

Question 7

Was using KANSEI for the first time easy ? If not, what would you suggest to help with this ?

The existing help document is clear enough. Training sessions are helpful, but might be too much for experienced researchers.
Question 8

Have you setup your own network bed and installed KANSEI on it? (if no, please skip questions 9-12)

No

Question 9

Are you satisfied with your installation of KANSEI? If no, Please elaborate

N/A

Question 10

Did you require assistance to install KANSEI on your own network bed?

If yes, What kind?

N/A

Question 11

What do you suggest we can do to help users in the future who want their own installations?

Yes. Installation of the Kansei software could be involved. Based on past experience, most users can install Kansei relatively easily within a week with help from OSU.
Question 12

How long did it take for installing and setting up KANSEI?

7 days

Question 13

Is KANSEI easily portable to a different architecture?

Not if the testbed to be configured is of a different architecture. Kansei now have a 3-tier architecture: server-stargates-motes. Having a drastically different architecture can complicate the installation of Kansei or even make it incompatible.

Question 14

Have you ever looked at the source-code of KANSEI? If yes, Why? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

yes

Question 15

Is the code well documented and easy to understand?

Most of it
Question 16

Have you ever needed customizations of KANSEI? If yes, please elaborate.

No

Question 17

Do you feel you can perform the necessary customizations in the future with your own group?

Depends on what type of customization. As long as the 3-tier architecture is maintained, customization should be relatively easy.

Question 18

Your comments on the quality of the documentation and user interface?

Does it need improvements?

It looks a bit dull now and a somewhat slow. The developers have been trying to identify the cause of the slow performance.

Question 19

Your comments on the reliability, availability, scalability, interoperability and performance of KANSEI?
It's readable, mostly available, scalable to a few hundred motes, interoperable with other testbed that are also use the Kansei software and reasonable performance.

Question 20

How did you hear about KANSEI?

I am one of the kansei developers

Question 21

Will you recommend KANSEI to others in the future? Why?

yes

Question 22

Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?

yes

A.1.6 Response 6:

Question 1*

What are you using KANSEI for?

Collecting data for Sensor networking protocols
Question 2

How long and how many and what kind of experiments have you used KANSEI for in the past?

A 5 years and a few hundreds of times may be. (Actually I have no clue, the number is just a estimate)

Question 3

Do you plan on using KANSEI in the near future? Do you see your usage increasing or decreasing?

Yes.

Question 4

Is there any period you use KANSEI more often (for eg: conference deadlines etc.)? If yes, is there any reason for doing so?

Yes. Usually during conference deadline and before demos. The reason is obvious, to get data for papers and to perfect the demo.

Question 5

Are you satisfied with your experience with KANSEI or does it need any improvements? If yes, in what way?
Yes and Yes. I am satisfied and it requires improvements. The ability to inject data into live experiments through the web interfaces has been planned and would be a good addition

Question 6

Have you encountered any problems with installing or while using KANSEI? If yes, please elaborate.

No.

Question 7

Was using KANSEI for the first time easy? If not, what would you suggest to help with this?

If a user is familiar with sensor networking programming, then Kansei is easy to use. If not familiar with sensor networking, (s)he has no purpose using Kansei

Question 8

Have you setup your own network bed and installed KANSEI on it? (if no, please skip questions 9-12)

I designed and developed Kansei for most part. So the question does not apply to me.
Question 13

Is KANSEI easily portable to a different architecture?

Yes and No. Kansei is designed for managing embedded systems of particular type. If the new user has the same type of devices its a piece of cake. If not, it could be ported, depending on how different the device is.

Question 14

Have you ever looked at the source-code of KANSEI? If yes, Why? and is the code well architected and implemented? And does the code need to be improved? If yes, Please explain.

Yes.

Question 15

Is the code well documented and easy to understand?

Yes.

Question 16

Have you ever needed customizations of KANSEI? If yes, please elaborate.

No.
Question 17

Do you feel you can perform the necessary customizations in the future with your own group?

Does not apply

Question 18

Your comments on the quality of the documentation and user interface?

Does it need improvements?

Documentation could be improved

Question 19

Your comments on the reliability, availability, scalability, interoperability and performance of KANSEI?

Reliability could be improved. Kansei has excellent scalability.

Question 20

How did you hear about KANSEI?

I designed and developed it.

Question 21

Will you recommend KANSEI to others in the future? Why?

Yes.
Question 22

Are there any open-source or commercial competitors to KANSEI that you know of that you considered or will consider?

Motelab

A.2 KANSEI DEVELOPER RESPONSES:

A.2.1 Response 1:

Question 1

What is the approximate number of Users of KANSEI?

No sense on this question. May be 20?

Question 2

Are there any equivalent competing open-source or commercial products?

as far as I know, there is no such software.

Question 3

Do users need training for using Ohio State’s setup of KANSEI?

Yes
Question 4

What is the approximate percentage of users who install and use KANSEI on their own hardware?

50%

Question 5

How much guidance or training do users need for adopting and using KANSEI on their own hardware?

Just read the kanseiGenie clone documentation should be helpful

Question 6

What percentage of the accounts are unused?

Not sure

Question 7

What percentage of the accounts is used just once?

Do not know

Question 8

How many users can use Ohio State’s setup of KANSEI at the same time?

may be 4-6? Did that number increase?
Question 9

Is active scheduling needed to avoid overlap of users request to use KANSEI in general or during an active period (near a conference deadline)?

Yes

Question 10

Are there any bug reports usually?

Yes, We have bug trac site.

Question 11

Is there an active development team which looks after the next updates of KANSEI or do they just maintain it (correcting a bug when reported etc.)?

Yes. OKgerms Team

Question 12

What is the approximate cost incurred for maintaining KANSEI?

Do not know

Question 13

What percentages of the users recognize or cite KANSEI in their papers?

a lot
Question 14

How much will it cost approximately to set up a small network bed and to install KANSEI on it?

do not know

Question 15

What percentage of the users request customizations?

not sure

Question 16

Are there any users who have worked on the code of KANSEI themselves for customization?

do not know

A.2.2 Response 2:

Question 1

What is the approximate number of Users of KANSEI?

20

Question 2

Are there any equivalent competing open-source or commercial products?
Yes, planetlab

Question 3

Do users need training for using Ohio State’s setup of KANSEI?

yes

Question 4

What is the approximate percentage of users who install and use KANSEI on their own hardware?

10%

Question 5

How much guidance or training do users need for adopting and using KANSEI on their own hardware?

very much right now, we hope to have livecd version soon

Question 6

What percentage of the accounts are unused?

70%

Question 8

How many users can use Ohio State’s setup of KANSEI at the same time?

Between 1 and 400
Question 9

Is active scheduling needed to avoid overlap of users request to use KANSEI in general or during an active period (near a conference deadline)?

No

Question 10

Are there any bug reports usually?

Yes

Question 11

Is there an active development team which looks after the next updates of KANSEI or do they just maintain it (correcting a bug when reported etc.)?

Yes

Question 12

What is the approximate cost incurred for maintaining KANSEI?

?

Question 13

What percentages of the users recognize or cite KANSEI in their papers?

?
Question 14

How much will it cost approximately to set up a small network bed and to install KANSEI on it?

Time largest expense, 0

Question 15

What percentage of the users request customizations?

0

Question 16

Are there any users who have worked on the code of KANSEI themselves for customization?

yes

A.3 JFTP DEVELOPER RESPONSES:

A.3.1 Response 1:

Question 1

How did this effort begin? Who were the first set of stakeholders?

Don't know
Question 2

Would you consider Jftp as a success? On what criteria do you base this conclusion?

Yes - broad usage achieved

Question 3

What started the adoption of Jftp in the community? How did people start adopting it?

Don't know

Question 4

How is this project sustained? Is it funded? If yes, how?

Don't know

Question 5

What do you believe are the different motivations for the multiple developers on this project?

Ego, and wanting something they can use themselves

Question 6
In what areas does Jftp need to improve? Give examples, specifically about the UI.

Native versus Java

Question 7

As there are multiple open source ftp clients available, what is the unique contribution of Jftp? What was the primary need that was met by Jftp?

Cross platform

Question 8

What is the collaboration/decision process for Jftp?

don't know

Question 9

How did the initial development team of Jftp form? How many developers were present initially when the project was started and how many developers were there towards the end of major development?

don't know

Question 10

Was the major chunk of development done by the initial team which started Jftp or were there a lot of contributions by the developers who adopted Jftp later?
A.3.2 Response 2:

Question 1

How did this effort begin? Who were the first set of stakeholders?

The FTP support in the JDK was awful, so I decided to implement the RFC myself.

Question 2

Would you consider Jftp as a success? On what criteria do you base this conclusion?

Absolutely. It has tons of downloads, I sold some commercial licenses and support (plus made contacts for some companies and have a good reference for my CV).

Question 3

What started the adoption of Jftp in the community? How did people start adopting it?

freshmeat.net, sourceforge

Question 4
How is this project sustained? Is it funded? If yes, how?

I work on it occasionally. But if somebody wants to contribute I can offer small bounties - just send me a mail.

Question 5

What do you believe are the different motivations for the multiple developers on this project?

Most contributors sent bugfixes, some had features ready and others improved the design - thanks especially to Jake K. (and all others of course) here!

Question 6

In what areas does Jftp need to improve? Give examples, Specifically about the UI.

There is some wasted space and the main action buttons should be bigger / better visible. A good UI designer can probably find tons of improvements.

Question 7

As there are multiple open source ftp clients available, what is the unique contribution of Jftp? What was the primary need that was met by Jftp?
Support for different APIs: SCP, SMB, ssh shell and utilities. Think about it that way: When a good HTML/JS-renderer is available it will become a browser, too.

Question 8

What is the collaboration/decision process for Jftp?

Talk to me ;)

Question 9

How did the initial development team of Jftp form? How many developers were present initially when the project was started and how many developers were there towards the end of major development?

Me initial, 3 active concurrently was the maximum I guess.

Question 10

Was the major chunk of development done by the initial team which started Jftp or were there a lot of contributions by the developers who adopted Jftp later?

There were very good contributions but I certainly did the most work.

Question 11

Is there anything you would like to let us know about the factors for the success and the sustainability of Jftp which we have not asked above?
- release early, release often - if you need devs, just ask on sourceforge - freshmeat.net provides good traffic and maintainers will find you - start the project when you have a working prototype ready, not too early

A.3.3 Response 3:

Question 1

How did this effort begin? Who were the first set of stakeholders?

I don't know.

Question 2

Would you consider Jftp as a success? On what criteria do you base this conclusion?

Jftp is a success, but it needs some more work for stabilizing and work with z/OS (MVS) systems.

Question 3

What started the adoption of Jftp in the community? How did people start adopting it?

I am a User of JFTP and supported the implementation of Z/OS functions with tests.

Question 4
How is this project sustained? Is it funded? If yes, how?

I don't know. I worked with any money.

Question 5

What do you believe are the different motivations for the multiple developers on this project?

The use in a multi-platform environment and easy use.

Question 6

In what areas does Jftp need to improve? Give examples, Specifically about the UI.

Jftp is a success, but it needs some more work for stabilizing and work with z/OS (MVS) systems. The problem is to get an access for the developers to a z/OS Host for testing.

Question 7

As there are multiple open source ftp clients available, what is the unique contribution of Jftp? What was the primary need that was met by Jftp?

It has a clear and easy gui, works with linux and windows. It is possible to transfer Data to z/OS -Host (like the windows-tool blue-zone-ftp).

Question 10
Was the major chunk of development done by the initial team which started Jftp or were there a lot of contributions by the developers who adopted Jftp later?

I don't know, I supported only tests.

A.4 JFTP USER RESPONSES:

A.4.1 Response 1:

Question 1

Are you satisfied with Jftp? In what areas can it improve?

Yes. It needs to improve on the UI part and add the drag and drop feature

Question 2

As there are many other FTP clients, what was the reason behind choosing to use Jftp?

Cross platform and looked simple enough to adopt if needed.

Question 3

What are the unique features of Jftp when compared to other open source ftp clients?

Supports multiple protocols and is cross platform simultaneously.

Question 4
Do you know what license is used by Jftp? Does the type of license influence your choice of open source software? Please explain.

No and No

Question 5

Have you ever requested features or reported bugs? How promptly does the development team respond to requests or bugs in general?

No

Question 6

What do you think of the quality of the code and documentation of Jftp?

Does it need improvements?

Code and documentation could be more clear.

Question 7

How did you hear about Jftp?

Sourceforge

Question 8

Would you recommend Jftp to others with similar needs?

Yes

Question 9
Is Jftp one of the best FTP clients you have ever used or do you prefer any other FTP client better? If so, which one?

Filezilla is better, but that is very complex as it uses C++

Question 10

How long have you used Jftp? Will you continue using it in the future?

1.5 years. Yes

Question 11

Have you ever contributed code to Jftp or other open source software? If yes, what factors do you consider before deciding to work on an open source software? (Few examples are programming language used, high scope for success, personal use, active development team, license, sponsor etc.)

Yes. Personal use and programming language used and the complexity of the software

A.4.2 Response 2:

Question 1

Are you satisfied with Jftp? In what areas can it improve?

Yes. Drag and drop capability.
Question 2

As there are many other FTP clients, what was the reason behind choosing to use Jftp?

Ability to use OSU ID to log into OSC.

Question 3

What are the unique features of Jftp when compared to other open source ftp clients?

Federated authentication between OSC and OSU.

Question 4

Do you know what license is used by Jftp? Does the type of license influence your choice of open source software? Please explain.

No. No.

Question 5

Have you ever requested features or reported bugs? How promptly does the development team respond to requests or bugs in general?

No.

Question 6

What do you think of the quality of the code and documentation of Jftp? Does it need improvements?
It could use more commenting.

Question 7

How did you hear about Jftp?

Sourceforge.

Question 8

Would you recommend Jftp to others with similar needs?

Yes.

Question 9

Is Jftp one of the best FTP clients you have ever used or do you prefer any other FTP client better? If so, which one?

FileZilla looks more polished and has a bigger installed base.

Question 10

How long have you used Jftp? Will you continue using it in the future?

2 years. Yes.

Question 11

Have you ever contributed code to Jftp or other open source software? If yes, what factors do you consider before deciding to work on an open source software? (Few examples are programming language used, high
A.4.3 Response 3:

Question 1

Are you satisfied with Jftp? In what areas can it improve?

Yes, I am satisfied.

Question 2

As there are many other FTP clients, what was the reason behind choosing to use Jftp?

jFTP does not require installation of software for functionality (i.e. no binary download or special permissions are required beyond those required to run Java).

Question 3

What are the unique features of Jftp when compared to other open source ftp clients?

Java-based GUI & console for entering SSH commands.

Question 4
Do you know what license is used by Jftp? Does the type of license influence your choice of open source software? Please explain.

No; the license is not my primary consideration and does not change the software's appeal IMHO.

Question 5

Have you ever requested features or reported bugs? How promptly does the development team respond to requests or bugs in general?

No.

Question 6

What do you think of the quality of the code and documentation of Jftp? Does it need improvements?

No, the quality is quite good and the documentation is consistent.

Question 7

How did you hear about Jftp?

I searched for jFTP.

Question 8

Would you recommend Jftp to others with similar needs?

Yes, I would recommend jFTP to others with similar needs.
Question 9

Is Jftp one of the best FTP clients you have ever used or do you prefer any other FTP client better? If so, which one?

jFTP is one of the best clients I have used (aside from the built-in 'ftp' command which is common to many Unix/Linux distributions).

Question 10

How long have you used Jftp? Will you continue using it in the future?

I have used jFTP for more than five years and will continue to use it.

Question 11

Have you ever contributed code to Jftp or other open source software? If yes, what factors do you consider before deciding to work on an open source software? (Few examples are programming language used, high scope for success, personal use, active development team, license, sponsor etc.)

I have not contributed code.