December 2006

Metrics to Support Open Source Software Adoption Decisions

Sanjeev Kumar  
Ross School of Business

Li Wang  
Ross School of Business

Follow this and additional works at: http://aisel.aisnet.org/amcis2006

Recommended Citation
Metrics to Support Open Source Software Adoption Decisions

Sanjeev Kumar  
Ross School of Business  
University of Michigan, Ann Arbor  
sankum@umich.edu

Li Wang  
Ross School of Business  
University of Michigan, Ann Arbor  
wang@umich.edu

ABSTRACT
Open Source Software ("OSS") is gaining popularity and the number of available OSS products is rapidly increasing. Increasingly business managers need to evaluate and select OSS products for adoption. However, OSS adoption presents unique risks and there is a need for metrics to assess these risks. In this research-in-progress we leverage publicly available OSS project information such as source code and CVS database to build a suite of metrics to help managers evaluate OSS products and assess OSS adoption risks. We also provide real project examples for calculation and interpretation of these metrics.

Keywords

INTRODUCTION
Open source software ("OSS") is developed by Internet-based communities of software developers who voluntarily collaborate in order to develop software that they or their organizations need (von Hippel and von Krogh 2003). OSS has become an important economic and cultural phenomenon. SourceForge.net, a leading infrastructure provider and repository for OSS projects, lists more than 100,000 such projects and more than 1,000,000 registered users (SourceForge 2006). Use of OSS products in firms has reached significant levels for many products and is growing at a rapid rate for many others. For example: Apache web server is estimated to run on 69% of all web servers in March 2006 compared to just 21% for the nearest competitor Microsoft (Netcraft 2006), while Mozilla Firefox browser achieved more than 10% market share within an year of its launch (Onestat 2006).

The open source phenomenon has attracted significant research interest but the focus has primarily been on the “supply side” of OSS. Although recent studies have started to focus on the “demand side” of OSS (Kumar and Krishnan 2005), considerable research gaps exist in our understanding of OSS adoption and use. In particular, practice press has been deeply concerned over challenges faced by managers to effectively select, evaluate, adopt and leverage OSS (Farber 2004; Goulde 2005). Many of these challenges are specific to OSS (e.g. licensing, unique support arrangements etc.) and managers need tools and techniques specific to OSS for taking informed decisions about OSS adoption.

In this research-in-progress paper, we argue that publicly available information about OSS including source code and development history in CVS\(^1\), can be leveraged to develop metrics that capture many of the risks of OSS adoption and hence help IT managers in OSS adoption decisions. While this research is expected to assist managerial decision making, it also aims to contribute to the extant research on IS adoption in general and OSS adoption in particular and improve our understanding of the “demand-side” of OSS. Although there have been initiatives to construct an Open Source Maturity Model (Navica 2006), we are proposing metrics that use publicly available information.

OSS ADOPTION CHALLENGES
IT managers face many challenges in successfully evaluating and adopting OSS products. Although academic research about these challenges is scarce, we can leverage practitioner literature to develop a holistic picture of these challenges.

\(^1\) Concurrent Versioning System. It contains the entire development history of the OSS project.
• Future development risk: A lack of future development and support guarantees is a significant risk factor in OSS adoption. As OSS developers are not under any obligation to continue the development, managers must factor in the risk of future unavailability of OSS products in their OSS adoption decisions. This risk, however, is related to the future functionality needs. If OSS is being considered for a commodity and stable functionality requirement then this risk is not significant. However, if the required functionality is expected to significantly change in future then the possibility of the OSS product not being developed or supported in future becomes a material risk. We can further detail the future development risk as follows:
  - Small group risk: If the developer group is small or most of the development is done by a small group of developers, then the risk that the project stalls in the future because of a few key developers leaving the project is high. Thus, small size of key developer group magnifies the future development uncertainty risk. Small developer group also implies that the peer-review mechanism of OSS for ensuring quality of development is also likely to be impaired, resulting in inferior products.
  - Product process combination risk: OSS is not suitable for all processes. Most of the OSS success stories like Apache, MySQL, Perl etc. have been in the “infrastructure” segment rather than in the “application” segment. IT managers need to decide which processes in the firm are suitable for which OSS product. Especially, managers need to match stable processes with mature OSS projects (even if future development risk is high) and dynamic processes with OSS products which are being actively developed and that have low future development risk.

• Other challenges: OSS adoption carries many other risks like licensing issues, legal issues, forking etc. However, in this paper we are focusing only on adoption challenges that can be assessed or mitigated using publicly available data about the OSS product.

Thus OSS presents many unique challenges to IT managers. However, OSS products also provide a lot more information about themselves that can be leveraged to assess the extent of the challenges outlined above.

FRAMEWORK FOR METRICS GENERATION

In this research-in-progress paper we are focusing on the three main sources for metrics generation: development pattern of OSS projects across time, distribution of development effort across developers and the structure of collaboration network of developers in the project. The completed research is expected to include metrics in other areas like design, reliability, quality etc.

Development patterns of OSS projects

The development history for OSS projects is publicly available and can be used to model the development curve of the project. Fig 1 shows the development curve for a real OSS project. The horizontal axis is the life span of the project and the vertical axis is the work done in the project (measured as lines of codes in this example). Both axes are standardized to a 0-1 scale for easier comparison across projects.

We have collected development history data for 100 OSS projects from SourceForge. We have observed that most of OSS development curves can be approximated by one of the four patterns shown in Fig 2 below.

We can see that pattern I and IV continue to have development momentum in the end and hence represent low future development risk. Pattern II and III represent those projects that have matured and are unlikely to change in future (high future development risk). Accordingly, projects with development pattern II and III are more appropriate when requirements are not expected to change in future.

Distribution of development effort across developers

Koch and Schneider (2002) showed that a small group of core developers do most of the work. Fig 3 below is a plot of cumulative percentage contribution by developers of a real OSS project (SourceForge Project No 27707). It clearly shows that only a few of the developers handle most of the work.

This distribution of work load across developers significantly impacts the future continuity of the development effort. Projects that depend on a few key developers are less likely to survive if a key developer leaves the project. Thus, a more

---

2 We have followed extensive quality control and sampling restrictions in collecting data from SourceForge. However, we are not detailing that here because of the lack of space.
equitable distribution of work between developers suggests lower future development risk. Equitable distribution would also enable more developers to be deeply involved in the project, thereby reducing the small group risk.

Figure 1: A Sample OSS Product Development Curve

(a) Type I Pattern: Time-limited Project
(b) Type II Pattern: Big Start Small End Project
(c) Type III Pattern: Normal Project
(d) Type IV Pattern: Multistage Project

Figure 2: Four Common OSS Development Patterns
Collaboration structure among developers

OSS development is a collaborative activity and the extent of collaboration between developers is likely to have significant impact on the current and future progress of the project. As OSS project’s CVS database provides information about which tasks (or files) each developer worked on, it is possible to draw a collaboration network of developers for the OSS project. These collaboration networks can be analyzed using social network analysis technique. In social network analysis, actors are modeled as nodes of a graph joined by their relationships depicted as edges (Wasserman and Faust 1999). Fig 4 below shows a collaboration network for a real OSS project (SourceForge Project No – 24184) where each node is a developer and each link shows the incidence of collaboration between two developers on a task.
A high level of collaboration implies that the project has shared skills among developers. Thus would make the developer network more resilient to shocks (such as a developer leaving a project) resulting in a lower future development risk. Similarly, high levels of collaboration would make it possible to leverage the developer group more effectively and hence represent lower small group risk as well.

**PROPOSED METRICS**

**Moment Metrics for capturing development pattern of OSS projects**

Let p denote percentage of project completed and t denote time taken. The development curve in Fig 1 can then be represented by a function \( p = F(t) \). The function \( F(t) \) is analogous to the Cumulative Density Function (“CDF”) as it starts from 0, never decreases and reaches its maximum value at 1. We can then define the corresponding Probability Density Function (“PDF”) \( f(t) \) as the first order derivative of \( F(t) \). The PDF represents the rate of development of the OSS project. Fig 3 shows an illustration of an example CDF and the corresponding PDF.

![Figure 5: CDF and the corresponding PDF for an OSS project](image)

Based on the PDF, four moment statistics: mean, variance, skewness and kurtosis, can be calculated for capturing the OSS development pattern. These statistics together can be used to infer the development patterns as shown in Fig 2.

- **First central moment - Mean (\( \mu \)):** Mean measures the average value of a random variable. Let \( f(t) \) be the PDF of random variable \( T \), then its mean is defined as:
  \[
  \mu = E[T] = \int_{-\infty}^{\infty} f(t) dt
  \]
  Since \( T \) can only take value between 0 and 1; and \( f(t) = 0 \) when \( t < 0 \) or \( t > 1 \), in our case,
  \[
  \mu = E[T] = \int_{0}^{1} f(t) dt
  \]

- **Second central moment - Variance (\( \sigma^2 \)):** Variance measures the spread of the distribution. In our case, variance can be defined as:
  \[
  \sigma^2 = E[(T - \mu)^2] = \int_{0}^{1} (t - \mu)^2 f(t) dt
  \]

- **Third central moment - Skewness (\( \gamma_1 \)):** Skewness measures asymmetry of a distribution. A skewed to the left PDF has a negative skewness while a PDF skewed to right has a positive skewness.
Fourth central moment – Kurtosis ($\beta_2$): Kurtosis measures the peakedness of a distribution. Large value of kurtosis corresponds to distributions with high peak while distribution with small kurtosis has a flat-topped probability density function.

\[
\beta_2 \equiv E[(T - \mu)^4] = \frac{\int_{-\infty}^{\infty} (t - \mu)^4 f(t) dt}{\sigma^4}
\]

OSS project development curves are not continuous as only discrete events are recorded in the project history. Therefore the development curve function $F(t)$ is discrete as well. Since the first-order derivative for the discrete functions does not exist, we need to approximate our definitions of moment metrics for the discrete case. If $d$ is the normalized time period between successive measurements then:

\[
\mu \approx \frac{\sum_{i=0}^{N} d_i f(t_i)}{N}; \sigma^2 \approx \frac{\sum_{i=0}^{N} d_i (f(t_i) - \mu)^2}{N} \quad \gamma_1 \approx \frac{\sum_{i=0}^{N} d_i (f(t_i) - \mu)^3}{\sigma^3}; \beta_2 \approx \frac{\sum_{i=0}^{N} d_i (f(t_i) - \mu)^4}{\sigma^4}
\]

All four moments can be calculated easily from OSS project histories in the corresponding CVS databases.

Empirical verification of moment metrics

We empirically tested the ability of moment metrics to correctly predict development patterns. We collected project history data for 100 projects from SourceForge.net. We then calculated the Probability Mass Function ("PMF", the discrete counterpart of PDF) and moment metrics for each project. The sample contained all four development patterns. Fig 6 shows the PMF plots for four selected projects, corresponding to the four typical development patterns. We then coded the project type of each project based on visual analysis of each development curve. Finally, we used Logistic Regression to test the relationship between moment metrics and the project type classification. Each logistic regression has a binary measure of development pattern as the dependent variable and the moment metrics as the explanatory variables. As there four development patterns, they give rise to four different binary measures resulting in four logistic regression estimations. Results of the regression are shown in Fig 7 below.

All four models were statistically significant, providing support to the argument that moment metrics can effectively capture development pattern of real project data. The results indicate that mean is a strong predictor for both type I and type II patterns while variance is a good predictor for type III pattern. Variance and kurtosis together are significant predictors for type IV pattern. Skewness does not appear to be significant predictor in these models because of its strong correlation with mean. Skewness becomes a significant predictor for both type I and II patterns if we take mean out of the regression.

Interpretation of moment metrics

Since the moment metrics are calculated based on standardized software development curves, IT managers can compare these metrics across different projects and then make their adoption decisions based on their requirements to minimize product process combination risks. Projects with large mean and large negative skewness are likely to belong to the type I development pattern; which is more suitable for stable processes. On the other hand, small mean and large positive skewness are characteristics for the type II pattern, which signify a project with growing functionality and low future development risk. Such projects are more suitable for processes with evolving and dynamic requirements. Both type III and IV projects are likely to have small skewness with mean values close to 0.5. These two patterns can be differentiated by the variance and kurtosis; type III projects tend to have smaller variance and larger kurtosis than type IV projects.

Determining the software development pattern is subjective and different people often have different opinions. The moment metrics provide a quantitative measure for these patterns and hence can help managers better assess the product process combination risks. Therefore these moment metrics can help IT managers select the appropriate OSS product for their needs.
Figure 6: Example PMF plots

Type 1: SourceForge Project No 120014

Type 2: SourceForge Project No 120008

Type 3: SourceForge Project No 120014

Type 3: SourceForge Project No 105505

Figure 7: Results of the logistic regression

<table>
<thead>
<tr>
<th>Type</th>
<th>Mean</th>
<th>Variance</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type I</td>
<td>24.71 (0.0004 ***)</td>
<td>-0.77 (0.94)</td>
<td>0.38 (0.17)</td>
<td>0.01 (0.32)</td>
</tr>
<tr>
<td>Type II</td>
<td>-15.53 (8.0e-5***)</td>
<td>-5.45 (0.54)</td>
<td>0.60 (0.13)</td>
<td>-0.04 (0.06)</td>
</tr>
<tr>
<td>Type III</td>
<td>0.48 (0.78)</td>
<td>-29.34 (0.012*)</td>
<td>-0.14 (0.42)</td>
<td>-0.007 (0.30)</td>
</tr>
<tr>
<td>Type IV</td>
<td>1.20 (0.7286)</td>
<td>17.51 (0.05*)</td>
<td>0.08 (0.91)</td>
<td>-0.86 (0.04*)</td>
</tr>
</tbody>
</table>
Development concentration metrics

Concentration of development effort can be measured using the popular Gini-coefficient (Yitzhaki 1979). In our case Gini-coefficient measures the inequality of effort distribution and ranges from 1 (high effort concentration) to 0 (equal effort by all developers). Gini coefficient for OSS projects can be calculated using the cumulative contribution plot shown in Fig 3. It is defined as a ratio with the area between the actual effort distribution (curved red line in Fig 8 below) and the ideal effort distribution (diagonal blue line in Fig 8) as numerator and the total area below the ideal effort distribution line as the denominator. Lower Gini-coefficient indicates more equitable effort distribution and lower future development risk and lower small group risk.

![Figure 8: Calculating Gini-Coefficient (SourceForge Project No – 27707)](image)

Developer collaboration metrics

Once the collaboration network is drawn (Fig 4), level of collaboration can be measured by the density of the network. Density is defined as the proportion of ties in a network relative to the total number possible (Wasserman and Faust 1999). Network density has previously been shown to positively impact collaboration and innovation diffusion (Abrahamson and Rosenkopf 1997). Higher network density indicates higher levels of collaboration and lower future development risk and lower small group risk.

CONCLUSION

OSS adoption presents unique risks but these risks can be effectively assessed using metrics calculated from publicly available information about OSS products. This paper presents a framework for such metrics and illustrates three such metrics. Moment metrics capture the development pattern of OSS projects that can be used to assess the future development risk and process product combination risk. Developer concentration metric (Gini-coefficient) and developer collaboration metric (density of developer collaboration network) provide further insights into the future development risk and small group risk. We are in the process of extending the metrics to other important risk factors like quality, reliability, design etc. We expect to present the expanded and finalized metrics suite at the AMCIS 2006 conference.

REFERENCES


