Modeling Success in FLOSS Project Groups

Justin M. Beaver
Oak Ridge National Laboratory
P.O. Box 2008, MS-6085
Oak Ridge, TN 37831
865-576-0327
beaverjm@ornl.gov

Xiaohui Cui
Oak Ridge National Laboratory
P.O. Box 2008, MS-6085
Oak Ridge, TN 37831
865-576-9654
cuix@ornl.gov

Jesse L. St Charles
Carnegie Mellon University
Wean Hall 1327
Pittsburgh, PA
412-268-5866
jstcharl@andrew.cmu.edu

Thomas E. Potok
Oak Ridge National Laboratory
P.O. Box 2008, MS-6085
Oak Ridge, TN 37831
865-574-0834
potokte@ornl.gov

ABSTRACT

A significant challenge in software engineering is accurately modeling projects in order to correctly forecast success or failure. The primary difficulty is that software development efforts are complex in terms of both the technical and social aspects of the engineering environment. This is compounded by the lack of real data that captures both the measures of success in performing a process, and the measures that reflect a group's social dynamics. This research focuses on the development of a model for predicting software project success that leverages the wealth of available open source project data in order to accurately forecast the behavior of those software engineering groups. The model accounts for both the technical elements of software engineering and the social elements that drive the decisions of individual developers. Agent-based simulations are used to represent the complexity of the group interactions, and the behavior of each agent is based on the acquired open source software engineering data. For four of the five project success measures, the results indicate that the developed model represents the underlying data well and provides accurate predictions of open source project success indicators.

Categories and Subject Descriptors

I.2.11 [**Multiagent systems**]. I.6.5 [**Model Development**]. I.6.4 [**Model Validation and Analysis**]. D.2.4 [**Software/Program Verification**]: Statistical Methods. D.2.4 [**Metrics**]: Performance Metrics, Process Metrics.

General Terms

Algorithms, Measurement, Design.

Keywords

Software Engineering, Agent-based Simulation, FLOSS, Data-based models, Bayesian Belief Networks.

1. INTRODUCTION

The complexity of a software engineering project is inherent given both the technical intricacies of code design and development, and the social aspects of working as a team. These difficulties are exacerbated in teams that develop Free/Libré/Open Source Software (FLOSS) by the distributed nature of the team, the lack of an organizational hierarchy, and the fact that source code contributors are often volunteers. Despite these challenges, there are a significant number of FLOSS projects that are highly successful in terms of their ability to deploy mainstream products. The Linux operating system [15], the OpenOffice application [16], the Mozilla suite of Internet tools [10], and the Apache web server [17] are all current examples of modern FLOSS success stories. These projects have overcome the social and technical challenges associated with FLOSS development and have consistently produced free software products that are competitive with their commercial counterparts.

This research seeks to understand the success of these projects in spite of such challenges. We propose a model intended to represent a FLOSS project by simulating both the behavior of each individual developer in the project, and a manifestation of those cumulative individual behaviors as group actions and decisions. The goal of this model is to analyze and predict success in FLOSS projects. FLOSS communities are environments that offer unique insight into the mechanics of the communication between individuals, the actions of software contributors, and the popularity and performance of projects. In the collaborative infrastructure of FLOSS websites, individual actions and communications are tracked and measured in detail, and can serve as direct inputs into the model of FLOSS software development. This research leverages that availability of data in order to create a data-based model of the complexity of FLOSS software development projects.

We propose a model that uses agent technology to simulate both a FLOSS project and each developer within that project in order to characterize it in terms of its complex social and technical interactions. The developed model is focused on the simulation of relatively large FLOSS development teams that have achieved a minimum threshold of team members. This problem space contains a project membership significant enough to include the social networking factors in the model. The FLOSS data that is freely available is used to both train the system's agents and also to validate the model in terms of its representation of the training data set and its predictive validity.

This research describes the model used to simulate FLOSS projects, and the results of validating that model, in an attempt to better understand the factors that influence success in software engineering projects. Section 2 reviews the relevant literature associated with the analysis of FLOSS projects and prior work in modeling these projects. Section 3 describes our methodology for

acquiring data, creating a model, and the architecture for implementing the model. Section 4 outlines the approach to validating the developed model including a description of the statistical tests employed. The results of the validation and an associated discussion are captured in Section 5 and conclusions are drawn in Section 6.

2. RELATED WORK

FLOSS development is a phenomenon in which software products are created largely through voluntary contributions from a distributed team of software developers. The software source code is freely available under various progressive licenses that permit users the flexibility to augment and modify the software as needed. The analysis of software engineering data associated with FLOSS projects has been the subject of many studies. This section explores the prior work related to the modeling, analysis and measurement of FLOSS software development projects.

This work is focused on developing a model of success in engineering FLOSS projects. However, there are several definitions of success in software engineering, each of which is centered on different priorities, such as the quality of the software, the efficiency of the process, or the effectiveness of the team. For FLOSS projects, we are interested in defining project success in terms of the dynamics of the distributed group. Leveraging the prior work described below, we focus on the ability of the distributed group to maintain or grow in membership, to effectively organize and coordinate source code contributions across multiple developers, and to produce software products that are useful in the user community. Success in a FLOSS project is demonstrated though those capabilities, and quantified through the indicator measures proposed Section 3.2.

Crowston, Howison, and Annabi have performed research on identifying metric indicators of success in FLOSS projects. In 2003 [7], they performed an analysis of FLOSS developer responses to a questionnaire in an attempt to determine the success factors that developers look for in projects. An anticipated result was that the majority of FLOSS developers gauged the success of a project based on their personal satisfaction in contributing to the project. The study also revealed that personal recognition and level of involvement of the users were driving criteria for developers in assessing a FLOSS project's success.

Crowston, et al., followed up their 2003 study with an empirical analysis of 122 FLOSS projects to identify four measures, extracted from the available FLOSS data, which were evaluated as indicators of project success [6]. The four measures included the size of the development team, the time to fix bugs, the number of downloads and the SourceForge activity level. In their analysis, the authors stress that a breadth of measures is necessary to completely characterize the success of a FLOSS project. They also identify three of the four measures as potentially good project success measures. The SourceForge activity level was deemed to be unsuitable as it is derived from the others and so is essentially redundant.

The social and behavioral aspects of FLOSS development have also been studied. Hahn, Moon, and Zhang [9] empirically examined the formation of FLOSS project teams and validated the significance of previous social ties on the probability of a project to attract developers. Ngamkajornwiwat, et al. [8], explored the evolution of FLOSS communities in terms of social networking measures. A social network is a graph comprised of nodes, representing individuals, and ties (or arcs) between the nodes that represent a relationship between individuals. Their results indicate that there are distinct convergent patterns in the social networks of FLOSS communities as projects evolve. Crowston, et al. [18] explored the relationship between software engineering metrics and the social network of projects in order to discern the core development team associated with a project.

While the above studies focus on the analysis of FLOSS data, very little work has been done to model the behavior in FLOSS projects. Crowston, et al. [19] proposed a model for effective work practices in FLOSS development. The model was based largely on an existing model of group effectiveness initially proposed by Hackman [20] in 1986. The model was a taxonomy of attributes and sub-attributes intended to serve as a normative reference for future analyses. While the paper provided no validation for the organization, it did outline several propositions, which form the hypothetical underpinnings for model validation in future studies. There does not appear to be work in the area of modeling the behavior of FLOSS groups in order to characterize or forecast a project's success.

We seek to address this gap in the research by developing a model of FLOSS projects in order to better understand the performance of those groups. In addition to modeling project-level decisions and actions, we propose an architecture where the behavior of each developer is modeled as a unique contributor capable of deciding the extent of the contribution. To our knowledge, no prior work has attempted to model FLOSS projects in order to get an improved understanding of emergent behavior within these complex systems.

3. TECHNICAL APPROACH

This section discusses the technical approach to the modeling of FLOSS projects. We begin by describing the criteria for FLOSS project selection and acquisition of data. Next, the design of the agent-based simulation is presented, including the model for the project environment. Finally, the social network elements of the model are presented including the approach for representing the decisions of humans.

3.1 Data Collection

This research has leveraged open source software development data freely available through the SourceForge Research Data Archive, hosted by the Department of Computer Science and Engineering at the University of Notre Dame [4]. This data repository has housed metrics for open source projects from the SourceForge website since April 2003. SourceForge [5] is an online center for FLOSS development communities, and provides collaborative resources for approximately 200,000 projects.

We developed scripts that query the SourceForge Research Data Archive for project data that meet our criteria. Because this project seeks to understand both the social and technical impacts on projects, establishing a minimum threshold for team size was necessary. We were interested in projects that reached a minimum team size of 20 developers at some point in the project lifetime in order to ensure that both the social and technical factors were well represented. In addition, we eliminated those projects that did not appear to use the SourceForge collaboration tools as a significant means for communication and coordination,

based on the message board and artifact statistics reported monthly for each project. In all, we identified 67 projects as viable for use as training data for our model.

3.2 FLOSS Success Measures

Forecasting the success of a software engineering project depends largely on the effectiveness of the selected measures as indicators of software development success. The FLOSS success measures used in this research are listed in Table 1. Each success measure is listed in terms of an abstract label, a specific metric used in the FLOSS domain, and a description of the interpretation of the measure.

Leveraging the analysis of FLOSS success measures performed in [6], we used Number of Developers and Number of Downloads as indicators of project success. We selected two additional SourceForge project metrics, Development Status and Group Ranking as indicators of success. Development Status provides visibility into the effectiveness, or maturity of the FLOSS Organization. Group Ranking is a measure of the popularity of a given project relative to other SourceForge projects. We also selected the Number of Releases as a measure of project success based primarily on the logic that an increasing release count is indicative of a flourishing organization. We believe these five measures are indicators of effectiveness and organization in a group. The goal of the simulation is to accurately represent the underlying FLOSS data in terms of the Table 1 measures, and also to reliably predict future trends in these measures for each studied group.

Table 1. FLOSS Success Measures

Success Measure	FLOSS Domain Metric	Description
Group Maturity	Development Status	Measures the efficiency and effectiveness of the group.
Group Membership	Number of Developers	A count of the group's core membership.
Number of Events	Number of Software Releases	A count of the number of orchestrated actions that the group has performed.
Group Utility	Number of Downloads	Measures the degree to which the group's actions are found to be useful in the community.
Group Popularity	SourceForge Group Ranking	The popularity of the group in the community.

3.3 Simulation Design and Implementation

Our approach to reproducing the complex environment of FLOSS software development was to use an agent-based simulation framework and model each project developer as a unique agent, called a Developer Agent, capable of making independent decisions. In addition, a software agent called the Project Agent was used to model the collective actions and decisions of the group. Figure 1 depicts the agent architecture used in this simulation. Software agents, depicted as circles in the architecture diagram, represent the behavior of both individual developers and also the FLOSS project as a whole. Connecting the agents are interfaces, shown as rectangles, which serve as the mechanisms for communicating project state to each Developer Agent, and for communicating individual developer contributions to the Project Agent. A developer contribution to a project is either a message posted, a software bug that has been fixed, or a section of original source code to increase project capabilities.

To run the simulation, the Project Agent and each of the initial Developer Agents initialize to either random values or known values, depending on the presence of any project-specific initial conditions. Information flows cyclically from the project to the individual agents and is fed back to the project again (as depicted with arrows in Figure 1) until stopping criteria are met. In this simulation, the stopping criterion was simply the number of time slices for which the operator wished to forecast the project's success. The Project Agent uses the developer contributions to make decisions about project-level events, such as the occurrence of a software release, or the addition/subtraction of project developers. The Developer Agents use the project-level events to make decisions about their individual level of contribution in terms of both source code and communication via project message boards. The simulation is implemented using the Multi-agent Simulator of Networks (MASON) [14], a set of libraries provided jointly by George Mason University's Evolutionary Computation Laboratory and Center for Social Complexity.

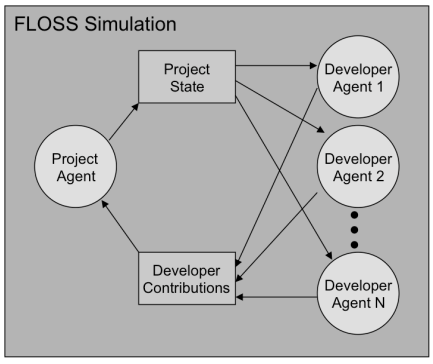

Figure 1 FLOSS Agent-based Simulation Architecture

The Project Agent was designed to be a data-based model that would exhibit behavior consistent with the collected project metrics. We selected Bayesian Belief Networks as the mechanism for modeling causal relationships between acquired metrics. A Bayesian Belief Network (BBN) is graphically represented as a collection of nodes that are connected by directed arcs. The nodes that comprise a BBN represent random variables within the model, with each node providing probabilities of outcomes based on a set of input values. The directed arcs in a BBN represent causal relationships or dependencies between nodes. The mathematical function that governs each BBN node is Bayes' Rule of conditional probability, originally proposed in [21]. A BBN represents chains of cause-effect relationships in which outcomes at each node are calculated as conditional probabilities based on the current state(s) of the node's inputs.

In order to identify a graphical structure for a BBN, causal relationships between the relevant random variables, the FLOSS metrics, had to be established. The first step in that process was to identify the set of random variables that could be modeled, and then determine those that were relevant. The design approach to the simulation and the metrics that were available from the SourceForge data framed the type of measurements that would be used as the inputs and outputs of the Bayesian model. The agents that represented individuals were constrained to produce measurement values that were representative of an individual's contribution to a project. There were several measures that were available, including the number of source code contributions, the number of messages posted, and the number of bug fixes implemented. Similarly, the project agents were constrained to produce measurements that were appropriate to group behavior and decisions. These were primarily the candidates for the success measures, and included such possibilities as the number of releases, the group size, and the group's development status. Once these inputs and outputs were determined, other metrics that might be relevant in the simulation were considered. Measures such as the time since the last release, a change in the licensing scheme, and average bug-fix time were considered. In all, we identified 19 potential measures for use in the Bayesian model.

Once the potential variables to be used in the Bayesian model of group behavior were identified, the goal was to determine those factors that were relevant, and develop a structure that represented the causal relationships between factors. Kan [22] listed three criteria for using empirical data to establish causal relationships: (1) cause precedes effect in time or logic, (2) two variables must be empirically correlated, and (3) any observed correlation is logical. In the case of a Project Agent, decisions from individuals (represented as outputs Developer Agents), such as level of participation in the project, influence the project-level behaviors. Thus, aggregated developer metrics must precede project decision and project effect metrics in the BBN. Similarly, variables that are explicitly group decisions, as described in Section 3.2, must be outputs of the group decision model. Thus, with Project Agent inputs and outputs defined, the dependencies between input and output nodes must be determined in addition to including any relevant intermediary variables.

In order to determine dependencies between Project Agent model variables, we initially analyzed the correlations between metrics. Although this revealed very few significant relationships, it did serve to eliminate some potential relationships between variables, thus reducing the state space. In the absence of significant correlations, we relied on the Mutual Information (MI) between variables, a technique for feature selection in machine learning [23], as an indicator of significance. MI is a measure of relative entropy between two random variables that quantifies the extent to which uncertainty in one random variable is reduced by knowledge of the state of the other random variable. The MI analysis further identified significant relationships between variables. The structure of the network was then developed by exploring different combinations of potential variable relationships in order to optimize the model's performance.

The Project Agent's Bayesian network is shown in Figure 2. We have partitioned the model structure into three tiers in order to more easily associate collections of random variables with their role in the simulation structure. Software engineering and project metrics from the Developer Agents are accumulated and entered as known values into the nodes at the Acquisition Tier of the network. Discretization of continuous input values was accomplished using Equal Width Discretization (EWD) [24], with each interval size based on standard deviation of the metric's underlying distribution. The six metrics identified in the Acquisition Tier were found to be the most significant factors in the MI analysis. The Project Decision Tier infers changes in the state of the project based on the inputs. These changes include whether or not the project is ready to make a software release, or whether the maturity of the project has improved. The Project Agent makes decisions on the state of each of these variables by applying a random number generator to the distribution of belief values associated with the appropriate node. The belief values are calculated by determining the joint probability distribution using the *a priori* SourceForge data and the current state of the nodes from the Acquisition Tier. The Project Effects Tier uses the inferred states of the Project Decision Tier to deduce the effects of those states on variables such as the number of downloads of the software product, or the group ranking (popularity) of the project. The states and associated beliefs of the Project Effects Tier are the data that drive the decisions made in the Developer Agents.

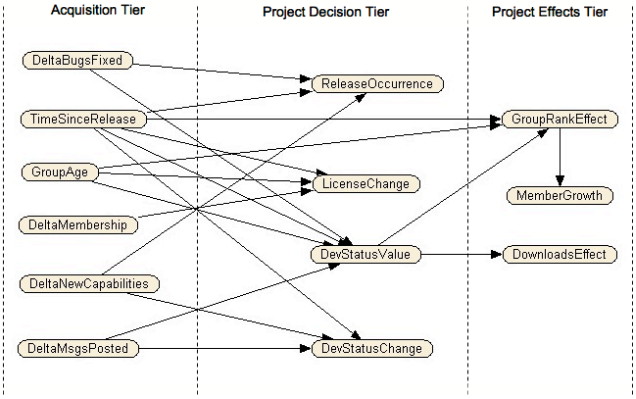

Figure 2 Project Agent Bayesian Belief Network Structure

At the core of each Developer Agent is a rule-based model that adjusts its behavior based on the current state of the project. The developed rules are based on heuristics, and are meant to reproduce each individual developer's degree of satisfaction with the project and motivation to continue to work on it. A Developer Agent's likelihood to contribute to the project in terms of forum messages, source code contributions, or bug fixes is dependent on that developer's perception of the project's progress, prestige, and utility. The Project Agent calculates and communicates values for project utility, popularity, and maturity to each Developer Agent, in addition to information regarding changes in project membership or whether a release has occurred. The rule set in each Developer Agent adjusts the probability that the developer will make a contribution to the project based on the current values of project-level information. The Developer Agent then determines a level of participation in the project by applying a random number generator to these adjusted probabilities, and communicating the type and level of contribution back to the Project Agent.

3.4 Social Modeling

The social element of the FLOSS project simulation was accomplished by modeling the Developer Agents and their interactions in a social network. Each Developer Agent is represented as a node in the social network, and social connections are represented as edges between nodes. For the purposes of this open source software research, a social connection between developers is defined as the co-occurrence of message posts in the same forum in a given project. The message posts, as acquired for each project through SourceForge, were a factor in determining the level of contribution of a developer to the simulated project.

4. MODEL VALIDATION APPROACH

Validating the proposed FLOSS model involves collecting and using a set of software engineering data, and using that data to confirm the ability of the model to characterize the data set, and the ability of the model to make predictions for unknown data. This section describes the approach to these activities including an outline of the validation process, and a description of the statistical tests and formulas used to quantify accuracy of fit and predictive validity.

4.1 Validation Process

The general approach to validation of the agent-based simulation of FLOSS projects is to train the model with the software engineering data and then measure the ability of the model to both accurately represent the distribution of selected measures for a given project, and accurately predict the distribution of those measures.

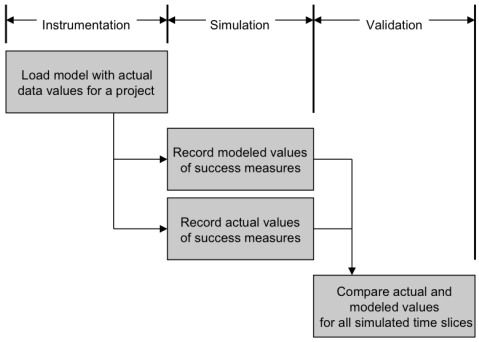

Figure 3 Model Validation Process

The validation process for the agent-based simulation involves three phases, and is depicted in Figure 3. Instrumentation is the phase where the model steps through a predetermined number of time-slices, but loads each of the random variables in the model from the actual data. In effect, Instrumentation is the initialization of the model to a specific project. In the Simulation phase, the model is run, and both modeled and actual data values for each of the random variables at each of the time slices are recorded. The Validation phase is the statistical comparison of the modeled and actual data. The specific tests that are used to validate the model are described in Sections 4.2 and 4.3.

In this validation exercise, the model was instrumented with actual project data for 9 time slices (i.e., 9 months of data), and then simulated for 1, 3, 6, 9, and 12 additional time slices. A "time slice" refers to a calendar month, which is the rate at which the counts for message post, source code contributions, etc. are calculated and recorded at the SourceForge Research Data Archive. At each interval, an assessment of the Accuracy of Fit and Predictive Validity were recorded. Each of the charts that present validation results use the projected (simulated) time slice values to show the trends for the Accuracy of Fit and Predictive Validity of each variable as the simulation progressed.

4.2 Determining Accuracy of Fit

The Accuracy of Fit measure is a quantitative determination of how well a model represents the underlying data. It is a measure that indicates the correctness of the model with respect to the data that was used to construct the model. Accuracy of Fit is determined through an analysis of the Equality of Means and the Equality of Variances between the modeled values of the open source software measures and the associated actual values.

The Equality of Means Hypothesis Test [3] (see Figure 4) quantifies the confidence that the mean value of two given populations are equivalent. By comparing the Equality of Means between actual open source software values and modeled open source software values, the ability of the model to accurately characterize the underlying data set is revealed. If the Test Statistic (t) for the given quality measure is less than the critical value ($t_{n-\nu,\alpha/2}$) for that measure, then the null hypothesis must be accepted, the means are determined to be equivalent, and the model is said to provide an accurate fit for the underlying data. For this study, a confidence of $\alpha = 0.9$, or 90% was used for all Equality of Means calculations. Thus, there is 90% confidence that all Equality of Means determinations are correct.

H_0: $\mu_{modeled} - \mu_{actual} = 0$ Reject H_0 if: $|t| > t_{n-\nu,\alpha/2}$

H_a: $\mu_{modeled} - \mu_{actual} \neq 0$

where, t = Test Statistic = $\dfrac{\mu_{actual} - \mu_{modeled}}{\sqrt{msE * (2 / n)}}$

and,

$\mu_{modeled}$ = the mean value of the modeled variable
μ_{actual} = the mean value of the actual variable
msE = the mean square error
n = the total number of samples
ν = the number of degrees of freedom
$t_{n-\nu,\alpha/2}$ = Student's t-distribution for confidence $(1 - \alpha) * 100\%$

Figure 4 Hypothesis Test for Equality of Means

The approach to calculating the Equality of Variances is to use a textbook rule of thumb test recommended in [1], where in comparing the modeled data to the actual data, if the ratio of the maximum variance value to the minimum variance value must be less than three to consider the variances equivalent. The application of this rule of thumb is appropriate in that it bounds the relationship of the variances. The goal for the Equality of Variances is not to get a quantifiable confidence on the accuracy of the modeled data (which is already accomplished through the Equality of Means test), but to get a discrete indication that the variance of the modeled data is on the order of the variance of the actual data.

4.3 Determining Predictive Validity

The Predictive Validity is a measure of how accurately the model predicts a variable using an unknown data set as input. It is a

quantitative way of determining how well a given model characterizes an unknown. Predictive Validity is measured using the Average Absolute Error (AAE) and Average Relative Error (ARE). AAE and ARE describe the deviations of the actual data from the modeled data, and are shown in Figure 5 and Figure 6 as defined in [2].

$$ARE = \frac{1}{N} * \sum_{i=1,N} \left| \frac{(y_{i,modeled} - y_{i,actual})}{y_{i,actual}} \right|$$

where,

$y_{i,modeled}$ = the modeled value of the variable
$y_{i,actual}$ = the actual value of the variable
N = the total number of samples

Figure 5 Calculation of Average Relative Error

$$AAE = \frac{1}{N} * \sum_{i=1,N} \left| y_{i,modeled} - y_{i,actual} \right|$$

where,

$y_{i,modeled}$ = the modeled value of the variable
$y_{i,actual}$ = the actual value of the variable
N = the total number of samples

Figure 6 Calculation of Average Absolute Error

By definition, the lower the values of AAE and ARE, the more closely the model approximates the actual data. The AAE and ARE in determining Predictive Validity provide assurance that the model developed is reliable in making predictions about unknown data. In this study, ARE was used to validate lower variance variables and AAE is used to validate higher variance variables, where values can confound ARE results. These measures have been used for the Predictive Validity of software engineering models in prior work [12][13], and in the case of ARE, a value of 0.25 or less (within 25% of actual value on average) is considered acceptable [11] to provide a useful prediction.

5. MODEL VALIDATION RESULTS

This section captures the results of applying the tests for Accuracy of Fit and Predictive Validity to an agent-based simulation that models group behaviors in the domain of FLOSS development.

5.1 Accuracy of Fit Results

The results of applying the Accuracy of Fit statistical tests, Equality of Means, and Equality of Variances are shown in Figure 7 and Figure 8. The Equality of Means analysis, shown in Figure 7, highlights the statistical threshold for this test as a black line. The interpretation of this graph is that those data points below the threshold line indicate that the simulation was able to maintain a mean value consistent with the underlying data for those metrics. Similarly, data points above the threshold indicate that the simulation diverged from the distribution upon which the simulation is based. Figure 7 shows that the mean values of four of the five FLOSS success measures were consistent with their actual mean values for up to three time slices. In the case of Group Membership, the mean value was consistent for nine simulated time slices. Thus the model performed well in remaining consistent with the underlying mean values of the data for short-term simulation, but became less consistent as the simulation progressed. The Group Popularity was an exception in this test, and is discussed further below. Only two measurement points are shown for Group Popularity to maintain a viewable scale for the figure, but the Test Statistic value continued to increase for 6, 9, and 12 time slices.

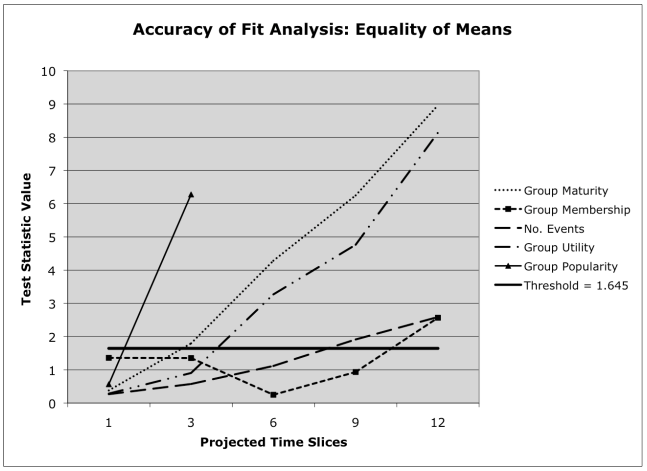

Figure 7 Equality of Means Validation

The results of the Equality of Variances test, shown in Figure 8, are similar to the Equality of Means in their interpretation. Values above the threshold line indicate points of unequal variance between actual and simulated data, and values below the threshold indicate points of consistency across the variances. The model did very well at simulating the variances for modeled values consistent with the underlying FLOSS data. The most striking exception, as with the Equality of Means test, is the Group Popularity measure.

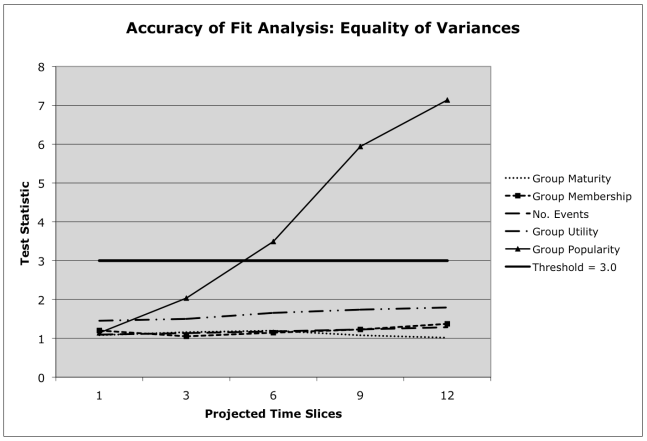

Figure 8 Equality of Variances Validation

For both of the Accuracy of Fit statistical tests, the model's lack of ability to represent the Group Popularity (SourceForge Group Ranking) measure is unexpected. The developed agent-based simulation performs well in modeling this measure for one time slice, and then its distribution quickly diverges for subsequent time slices. We postulate that Group Popularity is modeled

poorly because the assumptions regarding a group's influence on its popularity were flawed. In the agent-based simulation of FLOSS environments, each measure is calculated with the assumption that the actions and interactions of individuals in the group are causal factors in the emergence of group behavior. In the case of Group Popularity, this assumption does not hold because popularity is relative to the other groups in the environment. For example, Group Ranking, which is the SourceForge FLOSS measure for popularity, is relative to the popularity of the other projects on the web site. That is, it is not only the efficiency and organization of one group that affects its popularity, but that of all groups relative to each other. Because the current model is focused on the simulation of a single FLOSS group, it does not account for the presence and efforts of other groups. Thus, it is unlikely that it will be able to represent Group Popularity accurately. This is confirmed through the results of the Accuracy of Fit for that measure.

5.2 Predictive Validity Results

The results of applying the Predictive Validity tests, ARE and AAE, are shown in Figure 9 and Figure 10. The lower variance FLOSS success measures are shown in the ARE results in Figure 9. Recalling that an acceptable threshold for ARE is 0.25 or less, any values below the threshold line indicate that the model is a good predictor for that variable. The chart shows that for early predictions, two of the three variables are acceptable predictors. For both of those variables, the Number of Events and the Group Maturity, the model was a good predictor for up to three projected time slices. Group Membership was not as easily predicted in the earlier time slices, but converged towards the accuracies of the other variables at approximately the 6th time slice of simulation.

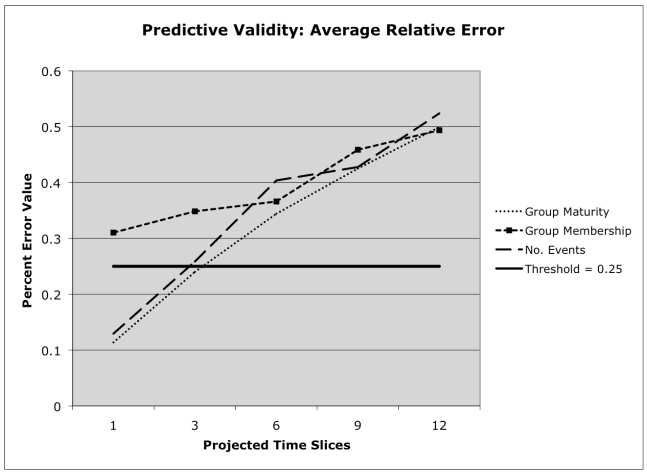

Figure 9 Average Relative Error (ARE) Validation

Figure 10 shows the AAE values for the high variance measures used to characterize FLOSS project success. These values are not normalized and so cannot be compared to a threshold. However, they can be analyzed in the context of their ranges. Group Popularity, for example, is instantiated in the model as the SourceForge Group Ranking measure, which has a range equivalent to the number of SourceForge projects (~200,000). So, an average error of approximately 5,000 for Group Popularity in the first time slice is surprisingly accurate given the range of the metric. Similarly, the Group Utility measure, represented in the model as the Number of SourceForge downloads, is relative to a range of tens of thousands. Similar to the success variables analyzed through ARE, the AAE values for Group Utility and Group Popularity indicate the model is a reasonable predictor of near-term future values, but its performance degrades significantly as the simulation progresses. More analysis is needed to refine the model such that its predictive performance can be improved.

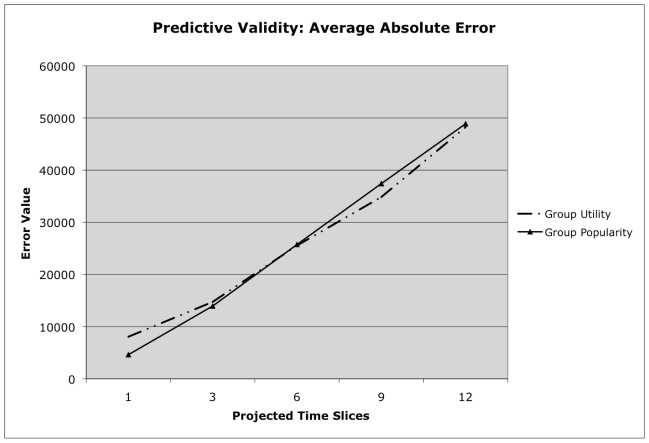

Figure 10 Average Absolute Error (AAE) Validation

The results presented in this section are the first steps in simulating the behavior of social groups using FLOSS data as a basis for modeling, and are promising as an initial attempt. We expect the Accuracy of Fit and the Predictive Validity to improve further as the model structure is refined and the scope of variables modeled is increased. The intent is to mature this model to be more robust and to represent a full spectrum of group and individual attributes that can be represented through real data sets when available.

6. CONCLUSION

We have developed a data-based model for group behavior that leverages FLOSS data and agent technology to produce a simulation that accurately represents the source data set. Using this model, we have been able to predict for group-level behaviors such as group membership changes, group efficiency and popularity, and the occurrence of group-level events or actions. We intend to extend this work to analyze the following:

- Model the self-organization of groups in order to identify those factors that contribute to a group attracting members,
- Explore in more depth the relative impact of each category of individual contributions on the project's success metrics,
- Model an environment with multiple groups that have low-coupling interactions, and
- Extend the model of individual developers to create a data-based model of individual behavior.

We believe the FLOSS environment provides a unique opportunity to quantitatively measure group behavior. Individual contributions are concretely tracked and measured in terms of both work products and social interactions. We intend to leverage

this environment to explore a data-based model for group behavior that is applicable in the context of software engineering, but also to a variety of other domains.

7. ACKNOWLEDGMENTS

This document was prepared by Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285; managed by UT-Battelle, LLC, for the US Department of Energy under contract number DE-AC05-00OR22725. This work was supported in part by the Office of Naval Research (N0001408IP20066). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the Department of Energy or the U.S. government.

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

8. REFERENCES

[1] A.M. Dean and D.T. Voss. Design and Analysis of Experiments. Springer-Verlag New York, Inc., New York, NY, 1999.

[2] T.M. Khoshgoftaar, B.B. Bhattacharya, and G.D. Richardson. "Predicting Software Errors, During Development, Using Nonlinear Regression Models: A Comparative Study." *IEEE Trans. Rel.,* **41**(3):390-395, September 1992.

[3] W. Mendenhall and T. Sincich. *Statistics for Engineers and the Sciences.* Prentice-Hall, Upper Saddle Ridge, NJ, 4th edition, 1995.

[4] University of Notre Dame. "SourceForge Research Data Archive (online)". https://zerlot.cse.nd.edu. Department of Computer Science and Engineering, University of Notre Dame. May 2008.

[5] SourceForge, Inc. "SourceForge Open Source Software (online)". http://sourceforge.net. SourceForge, Inc. 2008.

[6] K. Crowston, H. Annabi, J. Howison, and C Masango. "Towards a Portfolio of FLOSS Project Success Measures." In *Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering, International Conference on Software Engineering (ICSE) 2004,* pp. 29-33, 2004.

[7] K. Crowston, H. Annabi, and J. Howison. "Defining Open Source Project Success." In *24th International Conference on Information Systems,* 2003.

[8] K. Ngamkajornwiwat, D. Zhang, A.G. Koru, L. Zhou, and R. Nolker. "An Exploratory Study on the Evolution of OSS Developer Communities." In *Proceedings of the 41st Hawaii International Conference on System Sciences,* 2008.

[9] J. Hahn, J.Y. Moon, and C. Zhang. "Impact of Social Ties on Open Source Project Team Formation." In *Proceedings of the Second International Conference on Open Source Systems,* Como, Italy, June 8-10, 2006.

[10] Mozilla Foundation. "mozilla.org (online)." http://www.mozilla.org. Mozilla Foundation, 2008.

[11] S.D. Conte, H.E. Dunsmore, and V.Y. Shen. *Software Engineering Metrics and Models.* Benjamin/Cummings, Menlo Park, CA, 1986.

[12] J.M. Beaver and G.A. Schiavone. "Spatial Data Analysis as a Software Quality Modeling Technique." In *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering,* San Francisco, CA, July 2003.

[13] V. Shen, T. Yu, S. Thebout, and L. Paulsen. "Identifying error-prone software – An empirical study." *IEEE Transactions on Software Engineering,* **11:**317-323, April 1985.

[14] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. "MASON: A Multiagent Simulation Envoronment." *Simulation* **81**: 7, pp. 517-527, July 2005.

[15] Linux Online, Inc. "Linux Online! (online)." http://www.linux.org. Linux Online, Inc., 2008.

[16] CollabNet, Inc. "OpenOffice.org: the free and open productivity suite (online)." http://www.openoffice.org. CollabNet, Inc., 2008.

[17] The Apache Software Foundation. "The Apache Software Foundation (online)." http://www.apache.org. The Apache Software Foundation, 2008.

[18] K. Crowston, K. Wei, Q. Li, and J. Howison. "Core and periphery in Free/Libre and Open Source software team communications." In *Proceedings of the 39th Hawaii International Conference on System Sciences,* 2006.

[19] K. Crowston, H. Annabi, J. Howison, and C. Masango. "Effective work practices for FLOSS development: A model and propositions." In *Proceedings of the 38th Hawaii International Conference on System Sciences,* 2005.

[20] J.R. Hackman. "The design of work teams." In *The Handbook of Organizational Behavior,* J.W. Lorsch, Ed. Prentice-Hall, Englewood Cliffs, NJ, 1986, pp. 315-342.

[21] T. Bayes. "Essay Towards Solving a Problem in the Doctrine of Chances." *Philosophical Transactions of the Royal Society of London.* **53:**370-418, 1763.

[22] S.H. Kan. *Metrics and Models in Software Quality Engineering.* Addison-Wesley, 1995.

[23] I. Guyon and A. Elisseeff. "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research.* **3:** 1157-1182, 2003.

[24] J. Dougherty, R. Kohavi, and M. Sahami. "Supervised and unsupervised discretization of continuous features." In *Proceedings of the 12th International Conference on Machine Learning,* 1995.