Developer Behavior and Sentiment from Data Mining Open Source Repositories

William N. Robinson
Computer Information Systems Dept.
Georgia State University
wrobinson@gsu.edu

Tianjie Deng
Department of Business Information & Analytics
University of Denver
Tianjie.deng@du.edu

Zirun Qi
Computer Information Systems Dept.
Georgia State University
zqi1@gsu.edu

Abstract

Developer sentiment may wax and wane as a project progresses. Open-source projects that attract and retain developers tend to be successful. It may be possible to predict project success, in part, if one can measure developer behavior and sentiment—projects with active, happy developers are more likely to succeed. We have analyzed GitHub.com projects in an attempt to model these concepts.

We have data mined 124 projects from GitHub.com. The projects were automatically mined using sequence mining methods to derive a behavioral model of developer activities. The projects were also mined for developer sentiment. Finally, a regression model shows how sentiment varies with behavioral differences—a change in behavior is correlated with a change in sentiment. The relationship between sentiment and success is not directly explored, herein. This research project is a preliminary step in a larger research project aimed at understanding and monitoring FLOSS projects using a process modeling approach.

1 Introduction

Do developers react emotionally to the ebb and flow of a software development project? Open source software projects rely on developers that are motivated to donate their talents. Developers select projects based on the likelihood of success, among other factors[1]. Consequently, failing projects may be avoided or abandoned by open-source developers[2]. The perception of a project's success contributes to its success and is itself a measure of project success. Understanding developer sentiment toward a project may provide insight into their perception of a project's success. Therefore, we want to understand changes in developer sentiment that occur with changes in the software development process. Insight into this relationship may provide important project insights, including the behavioral-sentimental factors that lead to developers abandoning a project.

To illustrate how emotions play a role in project success, consider Yourdan's death march concept—a project that exceeds its normal parameters by 50% is a death march[3]. This typically occurs as resources are reduced relative to the planned requirements. For example, a project may have six months remaining on its schedule with a projected rate of completion in one year. Facing such a situation, managers may push developers to work harder. Some developers may even enjoy the hero's role as they strive to complete a project under difficult circumstances. Unfortunately, such projects typically fail.

Yourdan's death march applies mainly to commercial development, because developers are under contract to complete a project. In open-source development, when developers see the beginnings of a death march they are be more likely to simply abandon the project.

The death march illustrates an interesting issue about software development projects: due to the dynamic structure of the teams, and the unique ways of collaborating, projects can evolve substantially. Thus, analyzing a static snapshot does not allow us to understand an evolving trajectory. Evolutionary qualities are important, and may be a good guide to forecasting future qualities, such as project sentiment, followers, downloads, and eventual success

This research project is aimed at understanding the relationship between developer behavior and developer sentiment. This is part of the larger project that aims to create a developer behavioral-model to predict project characteristics, such as success. The ultimate goal is to provide automated support for project managers. As development occurs, the predictive model is continually updated and its results presented on a project dashboard.

Raising alerts on a project dashboard when a team appears to be losing its effectiveness is a goal of the work described herein. We rely on the team's repository to observe their activities. Our data mining techniques provide for automated analysis, which can be incorporated into the dashboards common to Agile development.

The automated approach to development monitoring and analysis is as follows:

(1) Developers use their standard development tools (e.g., Eclipse, GitHub, *etc.*) to develop software.
(2) The Monitor continually monitors events logged on the team's source code repository (e.g., GitHub).
(3) The Monitor builds predictive models, representing developer behaviors, from the repository events
(4) Analysis of the mined models is presented on a project status dashboard.

1530-1605/16 $31.00 © 2016 IEEE
DOI 10.1109/HICSS.2016.465

(5) Managers and developers review the analysis as a guide to how their team is performing.

This presentation is focused on step 3. Automation for the other steps has been described elsewhere (e.g., [4-9]).

1.1 Changing Behaviors

We aim to model developer behavior, developer sentiment, and the evolution of their relationship over time.

To acquire a model of developer behaviors, we data mine the event sequences contained in the software development repositories. We aim to identify and analyze common sequences of actions by developers. Developers commit code, raise issues, and make comments. We would like to know, for example, if the pattern sequence (Issue *x*, Comment *x*, Commit *x*) occurs frequently in the stream of developer activities. We suspect that successful projects include this pattern more frequently than simple *Commit x*—that is, having no previously specified issue or comment for a code commit.

To uncover sequential patterns, sequential data mining techniques are commonly applied. When applied directly to repository data, the techniques generate long lists of varied, low-level action sequences, which are not easily interpreted. To address this issue, we specify work constructs, which are recognized by a rule-base system. More generally, we are working towards a theory of FLOSS development by incrementally improving partial operational models, which recognize theoretical constructs in FLOSS event streams.

Our analysis proceeds as follows:

1. Specify recognition rules, derived from our theoretical constructs, such as an *issue-based work unit*.
2. Apply the rules to the repository event data, to recognize the theoretical constructs.
3. Data mine the recognized constructs, using sequence-mining techniques.
4. Apply model-differencing techniques, to recognize changes in behaviors over time.

This approach allows us to analyze developer behavior (as action sequences) and their changes over time.

1.2 Changing Sentiment

To acquire a model of developer sentiment, we apply common sentiment-analysis techniques. In general, the common approach is to use a dictionary lookup to tag words from a project as either negative or positive in their sentiment. The annotated words are then aggregated to arrive at a summary value indicating the amount of negative or positive sentiment. Our specific application of this general technique is described in section 3.6.

1.3 Process and Variance Methods

This project applies both process and variance methodologies to gain an understanding of the relationship between developer behavior and developer sentiment. A regression model is the main result we present herein; however, we apply process modeling to acquire some of the regression data.

A "process model explains development in terms of the order in which things occur and the stage in the process at which they occur."[10] Abbott, for example, illustrates how time ordered events affect the lifecycle of individuals, which supports theorizing about process steps, and cause-effect relationships[11]. Process models are useful in explaining IS changes [12, 13].

A process theory can be derived from data by analyzing event sequences. An event can be viewed as the change in state of some variable values[14]. An event sequence can be characterized by common metrics, such as length, entropy, subsequences, pattern frequency, and similarity to other sequences. Through abduction, a researcher can infer common constructs, higher level concepts, and eventually relate these terms to a theory that explains relationships among concepts within a set of boundary conditions[10].

The study presented herein uses a rule-based system to abstract low-level event sequences into an issue-based work unit, called a motif. Various process metrics are used to analyze the qualities and evolution of the motifs. Finally, these behavioral characteristics are compared with developer sentiment, using regression.

Although we have analyze each project's event sequences over time, we aggregate each project's behavior and sentiment data to create a multiple project regression, with behavior predicting sentiment. Regression is a variance model. Our analysis of 124 projects shows that sentiment changes with behavioral changes. In this way, our variance model summarizes the behavior-sentiment relationship as found in the 124 individually minded process models.

1.4 Article Overview

Next, this article introduces reference theories that have guided the constructs of our process and variance models. Section 3 presents the sequence mining GitHub projects. Section 4 presents the regression model. The final sections present a discussion and conclusions.

2 Background Theory

This section presents related theory on project attractiveness and sentiment and design routine theory.

2.1 Project Attractiveness

Project attractiveness plays a role in the relationship between developer behavior and developer sentiment, because sentiment and attraction are related. Developers with a positive project sentiment appear more likely to continue their work and encourage others. The FLOSS literature emphasize the importance of attracting users and developers to keep a project active and successful [15-18]. Developer motivation and participant interest has been suggested as an important factor

for success [19]. Some researchers also attribute FLOSS success to user interest [20]. Users, often serving as the observing "eye balls" to bugs [21], contribute to a project's success. Hence, it is important for a FLOSS project to attract both developers and users to be successful. Scweik *et al*. showed that for each developer added to an open source project, the chances of success increases 1.24 times[22]. Several studies attempted to identify what makes a FLOSS project favored by developers and users. Drivers of attractiveness include contributors' intrinsic and extrinsic motivations for joining FLOSS projects [23-27], contextual factors of the project [28], visibility of the project, and the work activities performed towards software maintenance and improvement [28].

2.2 Sustained Participation

A FLOSS project cannot survive without sustained participation. Success, which has been extensively examined in the FLOSS literature, is mostly measured at one time. Sustained participation, on the other hand, focuses on the long-run of FLOSS projects. Considering that 80 percent of FLOSS projects failed not due to the quality of their products, but insufficient long-term participation[29], it is crucial to predict a FLOSS project's sustained participation. Fang and Neufeld [27] investigated why developers continually contribute to FLOSS projects in a sustainable way. Results showed that situated learning and identity construction behaviors were associated with sustained participation. Qureshi and Fang [30] examined growth patterns of developers' socialization behavior and how that relates to their status progression. They identified four groups of newcomer behavior, based on the initial level of social resources of the developer and the growth rate of his/her socialization.

2.3 Sentiment Analysis of Development Issues

In summary, developers are attracted to, and participate, in projects for a variety of reasons. Their general feeling about a project may be considered their sentiment. When their sentiment turns sufficiently negative, they may abandon a project. It may be possible to infer a portion of their sentiment by analyzing their comments, posted in forums, issues, bugs, chats, and other means of communications.

GitHub.com is an example of a social coding environment[31, 32], which supports rich communications. Dabbish, et al. [33] found that developers use social coding capabilities for complex social activities, such as "inferring someone else's technical goals and vision when they edit code, or guessing which of several similar projects has the best chance of thriving in the long term. Users combine these inferences into effective strategies for coordinating work, advancing technical skills and managing their reputation."

We applied sentiment analysis of the GitHub.com issues, which provides a forum to track, and comment on, project issues. TABLE 1 summarizes prior research on sentiment analysis in software development and related domains. As can be seen from the table, sentiment analysis results are typically labeled with positive and negative polarity. To our knowledge, no prior research analyzed sentiment of project issues.

This project herein applies sentiment analysis to GitHub.com artifacts to infer developer sentiment. We compare changes in sentiment with changes in developer behavior. We infer developer behaviors from analysis of their repository actions, as interpreted by design routine theory.

TABLE 1 SENTIMENT ANALYSIS FROM PRIOR RESEARCH IN IS

Study	Data	Domain / Goal	Sentiment Analysis Technology	Result
[34]	Interviews	Software Development / Analysis Team Performance	Survey and Interviews	N/A
[35]	General web pages	General / Analysis sentiment in different context	Lexicon tagging, POS tagging, Semantic oriented approach	Polarity
[36]	Online forums	Finance / Prediction of stock index movement	Lexicon tagging, Classification	Polarity
[37]	Online consumer reviews	e-Commerce / Prediction of sales	POS tagging, Lexicon tagging, Clustering, Crowdsourcing	Polarity
[38]	Company internal emails	Text analysis / Classification of topic, opinion, style, genre	Writeprints, Ink Blots	Polarity
[39]	Online blogs	e-Commerce / Business Intelligence	Manually Classification	Polarity

2.4 FLOSS Development Routines

Routines are important to organizations because they are performed to accomplish work in organizations [40-43]. Routines and diversity of routines, have been commonly studied in both organizational literature[41, 44] and IS literature[45, 46]. Recently, Gaskin, et al. [46] argued the importance to study the variations of *design routines*. and defined a design routine as "a sequence of (design) tasks"[47].

2.5 Routine Diversity

Routine diversity has been referred by different terms such as routine variation [45-48], or routine heterogeneity[49, 50].

There are conflicting theories on the value of routine diversity. Pentland, et al. [48] believe that variation is "a prerequisite for change". Page argues that diversity can enhance robustness of complex and adaptive systems[51]. Diversity and variation is considered as a foundation for learning in general [52, 53] and for learning in routines [54]. Routine diversity allows actors to work in different ways, providing the flexibility required by dynamic environments. In the context of open source software development, Lindberg conducted a case study on Rubinius, an open source project, to investigate the co-evolution relationship between open source software development coding practice and communities [55]. A positive relationship between practice diversity and inflow of new developers was reported.

Most recently and most related to this study is the work of Robinson and Deng [56] who found that a moderate amount of design-routine diversity is associated with successful open-source software development. Success was measured as project stars (a kind of web bookmarks) and forks (i.e., copying). Herein, we analyze the relationship between design-routine changes and developer sentiment. This leads us to our hypothesis H1 of TABLE 2.

2.6 Routine Change

Routines are a central element of organizations and has been the subject of discussions on organizational stability and change [41, 44, 48, 57]. Pentland *et al.* suggests that routine changes are changes in "patterns of action" [48,p.1371]. Researchers hold different views on whether and how routines contribute to organizational stability and change. Some researchers conceptualize routine as stable, repetitive, standard operating procedures that do not change [42]. Others hold a "routine as change" perspective [41, 44, 48], viewing routines as "continuously changing entities"[57. page 171]. Pentland, et al. [48] argue that every performance of a routine is different due to the different time, places, actors and other objects involved. Levitt and March [54] attribute the routine change to direct organizational experience.

A project with unpredictability, such as a high level of temporal change, requires a substantial effort for participants to learn and adapt [58]. Dramatic changes in design routines increases the design difficulty, thus hinders participation [59-61]. Additionally, a high level of temporal change indicates a lack of control. Such projects will lose the capability of attracting new users and developers and their sustained commitment. This line of reasoning leads to the hypothesis H1 of TABLE 2.

2.7 Theoretical Propositions

From this literature, we have derived to two research hypotheses, shown in TABLE 2. These hypotheses are explored in the remainder. Note that both hypotheses refer to sentiment change, meaning a swing in either direction, positive or negative.

TABLE 2 MODEL HYPOTHESES

Hypothesis	Description
H1	Routine diversity is negatively associated with sentiment change
H2	Routine change is positively associated with sentiment change

3 Data Mining

Having introduced related work on design-routines and sentiment analysis, we now turn to our research project.

To analyze FLOSS design routines, we mine the event stream generated by developers as they use their tools. Developers have many interactions, directly or indirectly, through their tools. Some co-located developers will go to their computers to meet, thereby ensuring a record on their meeting (as well as providing access to development records).

Many FLOSS interactions are logged as event histories. For example, the history of source code changes is maintained by source control systems (e.g., CVS, Git). As each change is committed to the source repository, the new code and comments are recorded as a change event. Similarly, edits within a code editor (e.g., Eclipse), messages within a chat session, forum comments, feature requests, FAQ edits, *etc*. are all event sequences. Such sequences can be mined for patterns.

We analyzed 124 projects from GitHub, the most popular open-source code repository site. Founded in 2008, GitHub had over 3 million users and over 5 million repositories as of January 2013. After selecting projects, our KNIME workflow recognized design routines, generated behavior models based on the design-routines, and differenced those models. In addition, our workflow automatically ran sentiment analysis. We then ran a regression on the results of the data mining workflows, analyzing the relationship between developer behavior and developer sentiment.

3.1 Data Selection

We collected data from 148 FLOSS projects from GitHub. To obtain a sample with variation among successful projects, we used a stratified sampling strategy to sample projects with different level of popularity. The GitHub metrics, *number of stars* and *number of forks*, are proxies for the level of popularity to users and developers. We selected approximately 35 projects from each of the following sets:

1. >= 10,000 stars and >= 1,000 forks
2. 5,000 >= stars < 10,000, and 750 >= forks < 1,000
3. 1,000 >= stars < 5,000 and 500 >= forks < 750
4. 1,000 > stars and 250 > forks and in Java

We distinguished Java (in set 4) to investigate if language plays a role in projects' development patterns. (Most GitHub projects are scripting languages, like JScript, rather than traditionally complied languages.) The initial 148 projects

were reduced to the final 124 because some projects lacked sufficient data.

3.2 Data Preparation

A workflow automated the data acquisition and preparation. Data was obtained directly from GitHub.com and stored into a SQL database. The GitHub data is comprised of a 16 collections, which are combined, through filtering and joining, into a single table for data mining. Our data was derived mainly from these collections: issues, issue events, issues comments, pull requests, and pull request comments. Each record in the table provides a vector for input into our data mining process.

The table represents a sequence of Git events. Of the 18 Git events, we focused on six, which most closely associated with teamwork:

1. IssuesEvent: An issue is created, closed, or reopened.
2. PushEvent: Code is committed (pushed) to the repository.
3. PullRequestEvent: A user requests that new code be pushed to the repository.
4. IssueCommentEvent: A comment is associated with an issue.
5. CommitCommentEvent: A comment is associated with a commit (PushEvent).
6. PullRequestReviewCommentEvent: A comment is associated with a PullRequest.

Other events, such as watch events, in which users subscribe to a repository to get updates, do not concern teamwork. They are thus excluded from our study. Our prepared table of sequential Git events is further process to represent elements of design routines.

3.3 Design Routine Constructs

Git events, such as push and commit, represent work; however, the context of the work is missing. For example, it seems that 10 code commits for the same issue is different than 10 code commits, each for a single issue. A rule-based system is applied to the prepared event data to derive a table of abstracted work events.

Work in most GitHub projects begins with an IssueEvent or a PullRequestEvent. Both represent a typical unit of development work, which may be scheduled, opened, closed, reopened, *etc*. An IssueEvent typically represents a bug or enhancement. It follows a common lifecycle of being opened, followed by code changes represented by commits, and then an issue close. For example:

IssuesEvent.open, PushEvent, PushEvent, IssuesEvent.close

Of course, other events may intervene (e.g., comment events), as well as the issue may be reopened or never closed.

The PullRequestEvent is similar to the IssueEvent, but the subsequent work events are related to integrating the new code into the project's code repository.

A rule-based system is applied to recognize event sequences beginning with IssueEvent or a PullRequestEvent. We think about them as design routines, which are initiated in response to a work request (e.g., issue or pull request). However, we use the more neutral term, *motif*, to indicate recognition of these common sequence patterns.

The rule-based system recognizes two kinds of work motifs in the prepared table of sequential Git events. The basic form is as follows:

1. (IssueEvent | PullRequestEvent) .*
2. (Reopen (of #1)) .*

As indicated above, a work motif begins with either an IssueEvent or PullRequestEvent, followed by any other Git event that references the initiating event (by number). The motif records the initial event, and all subsequent events (and their attributes). When either an IssueEvent or PullRequestEvent is reopened, it is consider a new instance of the second motif pattern (above). Thus, open and reopen are each considered the beginning of a work motif.

These work motifs are derived from the prepared data in support of our theoretical background—tasks of distributed cognition in particular. Thus, we call these derived, abstracted elements work *constructs*, to be consistent with theorizing process theories[10].

3.4 Sequence Feature Construction

Before the work motifs can be sequence mined, they are encoded. Most sequence data-mining algorithms process event sequences, where events are identified as members of a fixed alphabet. An event sequence, for example, could be A-B-A-B-B-C. Few algorithms directly address object sequences, where each sequence member is comprised of an information object. An object sequence, for example, could be [Issue.ID=1,Issue.Author=a1,Issue.State=open]-[Commit.ID=10,Issue.Author=a1,Commit.IID=1]-[Issue.ID=1,Issue.Author=a1,Issue.State=close]. Object sequences can be processed by dropping information. For example, just processing object type information: Issue-Commit-Issue. More information can be processed by first encoding the object information into another alphabet; for example, [Issue.ID=1,Issue.Author=a1] becomes I1A1. We applied this transformation concept to the sequence of work motifs.

Given the work-motif sequences for a GitHub project, we using k-means clustering to transform each work motif into one of 50 clusters. (Size 50 clusters retained the information represented by the great variety of sequences, as indicated by cluster metrics, while simplifying the subsequent sequence analysis.) This construct is a *typed motif*.

The *typed motif* is represented by a vector of these attributes:

IssueEvent | PullRequestEvent (open|reopen), ID, openDate, events, actors(same|different), number of actors, state(open|closed), merged(true|false), duration, number of comments, turbulence. Our rationale for selecting those attributes is: different instantiations of the same work motif might present different levels of collaboration. For example,

in an IssueEvent→CommentEvent→PushEvent work motif, the level of collaboration would be different between an instantiation in which the same actor performed these three activities, and an instantiation in which three different actors performed these three activities. Distributed cognition theory, which considers how team members collaborate, seems to be an appropriate lens to interpret these data. Thus, we used concepts from this theory to extract relevant attributes of a work motif. We reviewed the indicators of distributed cognition based on existing literatures [62-64]. From these indicators, we identified relevant attributes in for the typed motifs.

3.5 Sequence Modeling

Given sequences of typed motifs, our software constructs models of (a) sequence pattern probabilities and (b) changes in the patterns over data windows.

Stream mining can detect changes in the data-stream. One method is model differencing, which provides a mean to recognize important changes occurring within an event stream. Consider a stream of repository events divided into data windows, (w_1, w_2, …, w_n). Transition identification marks each data window as either being normal or transitional; for example, (normal, normal, transitional, normal, normal …). Transitional behavior is historically unusual behavior, according to some measure such as statistical variance. We use the term *transitional* because the behavior is unusual and transient, and thus interesting from a theoretical perspective, such as cognition or learning theory.

In our approach, a repository stream is divided into data windows. Each window is characterized by a model, ($w_i \Rightarrow \lambda_i$). Consider two models in sequence, λ_1 and λ_2. The software finds the difference of the models to characterize the change: $d\lambda/dt = (\lambda_2 - \lambda_1) / (t_2 - t_1)$. If the difference $\Delta\lambda$ is significant, by some measure, then we have found a transition point[4].

In this work, the model types (λ) vary; however, we mainly apply hidden Markov models (HMMs). A hidden Markov model (HMM) is a stochastic signal model[65]. In our application to repository analysis, the signals are sequences of discrete typed events (e.g., code commit). Given event sequences, a common task is to find transition probabilities. That is, given an observed event A, what is the probably that the next event observed with be B or C? A hidden Markov model (HMM) can solve this problem by building a probability model from observed event sequences. We use HMMs to model patterns of typed motifs (i.e., design routines) within the stream of FLOSS repository events.

HMM transition identification detects significant changes in modeled events between consecutive windows of event data. HMMs can be used to identify transitions by: (1) *comparing consecutive HMMs* generated from the observation sequences[66], or (2) *comparing consecutive acceptance probabilities*[4]. This difference is known as ΔHMM.

3.6 Sentiment Analysis

For sentiment analysis, we adopt NRC Word-Emotion Association Lexicon v0.92 (NRC Lexicon) as keyword matching reference to perform sentiment analysis[67]. NRC Lexicon has 14,182 unigrams (i.e., single words), which are categorized into both polarity (positive/negative) and eight types (anger, anticipation, disgust, fear, joy, sadness, surprise, and trust). It was created manually by Amazon Mechanical Turk, which is a crowdsourcing platform.

Using a KNIME workflow, the NRC Lexicon was applied got issues and their associated comments, generating counts of positive and negative words for each data window analyzed. These are the same data windows analyzed for the typed motifs (i.e., design routines).

4 Data Analysis

Our research model proposes a negative relationship between routine diversity and sentiment change, and a positive relationship between routine change and sentiment change. These two hypotheses are summarized in the prior of **TABLE 2**.

4.1 Dependent variables

We use the average change ratio between the number of positive words and the number of negative words, as a dependent variable to measure the average sentiment-change magnitude of a project. To obtain it, we calculate the ratio between the number of positive words and the number of negative words in all the posts in every data window. Then we calculate the absolute differences of the ratio between every two consecutive windows, this gives the magnitude of sentiment-change, for a sequence of data windows—their average is obtained as the dependent variable.

4.2 Independent variables

Two variable constructs measure routine diversity and routine change:

1. Entropy, measures the uncertainty of different activity types; it indicates the diversity of the design routines.
2. ΔHMM, measures the magnitude of change in transition probabilities; it indicates the degree of design routine change.

Average entropy is an independent variable for each project. We used Shannon-Wiener index [68] to measure the average process diversity in a project. Shannon's index has been used to calculate entropy, which has been defined as a transversal distribution of activities[69]. In our context, entropy is the distribution of different work routines. A data window dominated by a single type of work routine will have low entropy, whereas a window characterized by diverse work routines will have high entropy. After preliminary analysis, we choose four-weeks for our data window size—it contains sufficient data and

represents a common unit of work for open source development methodologies. Entropy is calculated for each window to provide a sequence of project entropies over time—their average is used for the regression.

4.3 Control variable

A variety of characteristics can affect sentiment change. For example, the intensity of conversations (number of words exchanged), might have an impact on the magnitude of sentiment change. In addition, the number of people engaging in the routines might also affect sentiment change. Therefore, we control for average number of words per time window, and the average number of actors per routine per time window. The control variable definitions are in **TABLE 3** and the dataset descriptive statistics is in **TABLE 4**.

TABLE 3 VARIABLES AND DESCRIPTION

	Variable Name	Definition
Dependent	Δ Positive / Negative Words Ratio	Average number of positive words/number of negative words ratio difference
Inden	Entropy	Average uncertainty of event distribution in a given time period
	ΔHMM	Average HMM difference of the project
Control	Word Count	Average number of words per time window of the project
	Number of Actors	Average number of actors per routine per time window of the project

TABLE 4 DESCRIPTIVE STATISTICS

Variable	Mean	Std. Dev.	Min	Max
Δ Positive/Negative Words Ratio	0.82	0.63	0.14	4.12
Number of Actors	2.25	0.28	1.1	3.09
Word Count	5,692.7	9,384.2	67.5	71,148.1
ΔHMM	1.70	0.73	0.23	3.97
Entropy	0.89	0.09	0.48	1.00

4.4 Regression Analysis

We applied ordinary least-squares (OLS) regression to estimate the dependent variables: the average change of ratio between number of positive words and number of negative words per window. Before completing our analysis, we checked assumptions of the multiple linear regression (i.e., normality and multicollinearity among independent variables). We also conducted Shapiro-Wilk test to ensure the assumptions of normal distribution of residuals are not violated.

4.5 Regression Results

TABLE 5 presents the regression results. It shows a positive relationship between HMM difference and positive/negative words ratio difference, at the 0.001 significance level. However, entropy is not significantly associated with Δ positive/negative words ratio. Additionally, variables that significantly affect sentiment change (Δpositive/negative words ratio) include word count and number of actors. Thus, there is strong statistical support for the proposed relationship between routine change and sentiment change. H2 is supported but H1 is not supported.

TABLE 5 REGRESSION RESULTS

Variable	Positive/Negative Ratio Difference	
	Model1 (adj R^2=0.47)	Model 2 (adj R^2=0.58)
Constant	1.413***	-0.098
Word Count	-5.046E-005***	-2.644E-005***
Number of Actors	-0.692***	-0.429**
Entropy		-0.583
ΔHMM		0.416***

Note: *: P ≤ 0.05. **: P ≤ 0.01. ***: P ≤ 0.001

4.6 Intra-project Windowed Analysis

To reveal the implications of the project-level regression of the prior section, we review a sample project over time. **FIGURE 1** illustrates ΔHMM for sample projects. The *x*-axis represents the Kullback-Leibler comparison of HMMs generated from the data windows. Each point represents the comparison between two HMMs, each representing a month of data. The trend values are more important than the specific HMM comparison values. Notice that all projects have periods of transition, where their behavior models change significantly, as shown by the spikes. This figure illustrates how ΔHMM discriminates unusual periods of sequential behaviors from the more common background.

The transitions (spikes) displayed in **FIGURE 1** represent real changes in developer behavior—the developers have changed their patterns in typed motifs (i.e., design routines). We have correlated those changes with web data to validate that interesting behavioral changes occur at the spikes. Moreover, interesting sentimental changes occur near the spikes. This intra-project comparison on data model and web data validates the regression results.

Table VI illustrates antirez/redis project text that correlate to the transitions presented in **FIGURE 1** (The *x*-axis is data window count, not date.) These entries illustrate

corroborating evidence demonstrating that the transitions capture behavioral changes.

We can also compare the month (data window) values for ΔHMM with ΔPositive/Negative Word Ratio using Pearson's correlation. For redis, the Pearson's correlation is 0.53 for 53 data points, which indicates a strong positive correlation. Taken together, the project-level regression, with the data window-level Pearson correlation, and the web blog data all point to supporting *routine change is positively associated with sentiment change* (H2).

TABLE VI BLOG ENTRIES FOR REDIS TRANSITIONS.

Data window	Text events
Week 44 –45 (2013 06/02/2013 – 06/30/2013)	(1) Pull request #1165 was posted in June 2013, with the name "fix sentinel parse error in sentinelRefreshInstanceInfo". (2) On June 19, the author commented: "it breaks sentinel INFO parsing routine in sentinelRefreshInstanceInfo. So I fixed it." (3) June 20, the repository owner commented: "Hello @charsyam, ***good catch***! However I can't use the commit as it is (see my commit implementation in a few minutes for hints) but I'll surely ***credit*** you in the commit message." (4) The author replied with a ***smiley face***: "@antirez Yes, I know. maybe redis will needs more general parsing code for this ***:)***"
Week 50 –51 (2013 11/17/2013 – 12/15/2013)	(1)On Nov 8, 2013, malviyarahul2001 posted: "Hi, My redis slaves have suddenly started ***dying*** and I cant figure out why. There is no change in system configuration or hardware wise. Here is the dump from redis before it ***dies***." (2) On Nov 14, 2013,adilbaig commented: "I'm having the same ***issue***. I purged the rdb file on the slave and restarted the sync. Same issue. I'm on Ubuntu 12.04.3 LTS x64." (3) On Nov 20, 2013, adilbaig commented: "This has happened several times to our in-production servers, today it was several times a day." (4) On Nov 20, 2013, the owner antirez commented: "Hello, what happens is very ***strange*** and sounds like hardware ***instability*** at a first glance, because there are two crashes in two unrelated positions. The master is running on a different physical host and is stable?"
Week 65 –66 (2015 1/11/2015 – 2/8/2015)	(1) On Jan 28th, 2015, Robot7 posted: "So Antirez, you are going to delete facts now?" (2)Robot7 then went on commenting: "So Antirez, you made a response and the moment I replied to it with facts you decided to delete the issue? That is just ***not professional***. Github was showing 404, but I reloaded a few times and got the unflush cache from a github slave. Here it is, would you like this done by a bot from now on?"

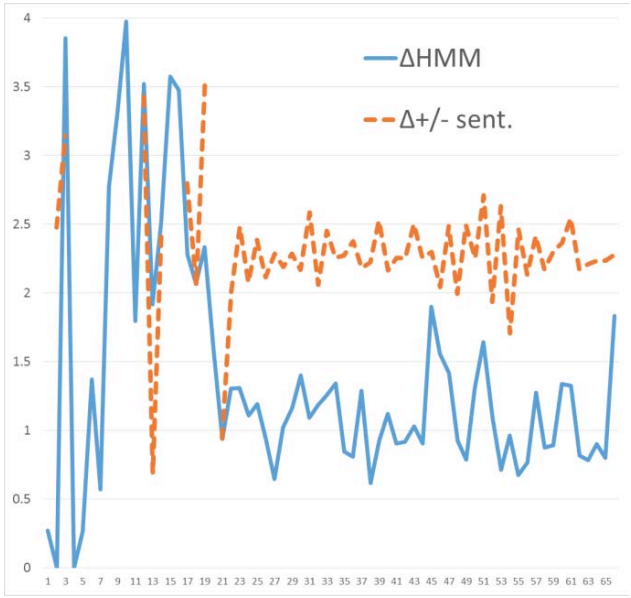

FIGURE 1 ΔHMM AND ΔSENTIMENT FOR ANTIREZ/REDIS.

5 Discussion

This study aims to understand the relationship between design-routine changes and developer sentiment. A multi-project level regression supports the conclusion that *routine change is positively associated with sentiment change* (H2). However, H1, that of *routine diversity is negatively associated with sentiment change*, was not supported. Nevertheless, it warrants further analysis.

In the context of FLOSS development routines, entropy measures routine diversity, while ΔHMM measures the magnitude of change. It would seem that active developers are more sensitive to both measures, while more passive people (e.g., users and occasional developers) are less sensitive. The lack of support for H1 may be attributed to a lack of concern about routine diversity. It is certainly worth further exploration.

Automation was implicit in our discussion; however, all the steps, from project data retrieval through the regression analysis are automated. Retrieval of the project records is, by far, the slowest part of the analysis. On an ongoing basis, a dashboard can update many projects every minute, which may benefit practitioners as they attempt to control their projects.

For future research, we propose to investigate intra-project correlations among our constructs, as illustrated in section 4.6. Limitations of this study include: (1) sampling, in that the projects may not be representative of FLOSS projects in general; and (2) modeling, in that the measurement of and constructs for sequential behaviors are incomplete. Future work will seek to diminish these limitations.

6 References

[1] I. Chengalur-Smith, A. Sidorova, and S. Daniel, "Sustainability of free/libre open source projects: A longitudinal study," *Journal of the Association for Information Systems,* vol. 11, pp. 657-683, 2010.

[2] I. Samoladas, L. Angelis, and I. Stamelos, "Survival analysis on the duration of open source projects," *Information and Software Technology,* vol. 52, pp. 902-922, 2010.

[3] E. Yourdon, *Death march*: Pearson Education, 2003.

[4] W. N. Robinson, A. Akhlaghi, and T. Deng, "Transition Discovery of Sequential Behaviors in Email Application Usage Using Hidden Markov Models," in *Hawaii International Conference on Software Systems*, HI, USA, 2013, p. (best paper nominee).

[5] W. N. Robinson, A. R. Syed, A. Akhlaghi, and T. Deng, "Pattern Discovery of User Interface Sequencing by Rehabilitation Clients with Cognitive Impairments," in *Hawaii International Conference on Software Systems*, HI, USA, 2012.

[6] W. N. Robinson, A. Akhlaghi, A. Syed, and T. Deng, "Discovery and Diagnosis of Behavioral Transitions in Rehabilitation Patient Event-Streams," *ACM Transactions on Management Information Systems (TMIS),* 2012.

[7] S. W. Hansen, W. N. Robinson, and K. J. Lyytinen, "Computing Requirements: Cognitive Approaches to Distributed Requirements Engineering " in *Hawaii International Conference on Software Systems*, HI, USA, 2012.

[8] W. N. Robinson, "A Roadmap for Comprehensive Requirements Monitoring," *IEEE Computer,* vol. 43, pp. 64-72, May 2010.

[9] W. N. Robinson, "A requirements monitoring framework for enterprise systems," *Requirements Engineering Journal,* vol. 11, pp. 17-41, 2006.

[10] A. H. Van de Ven, *Engaged Scholarship: A Guide for Organizational and Social Research: A Guide for Organizational and Social Research*: Oxford University Press, 2007.

[11] A. Abbott, "A primer on sequence methods," *Organization Science,* vol. 1, pp. 375-392, 1990.

[12] L. B. Mohr, *Explaining organizational behavior* vol. 1: Jossey-Bass San Francisco, CA, 1982.

[13] K. Lyytinen and M. Newman, "Explaining information systems change: a punctuated socio-technical change model," *European Journal of Information Systems,* vol. 17, pp. 589-613, 2008.

[14] K. Chandy and W. R. Schulte, *Event Processing: Designing IT Systems for Agile Companies*: McGraw-Hill Osborne Media, 2009.

[15] R. Y. Arakji and K. R. Lang, "Digital consumer networks and producer-consumer collaboration: Innovation and product development in the digital entertainment industry," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, 2007, pp. 211c-211c.

[16] S. Koch, "Profiling an open source project ecology and its programmers," *Electronic Markets,* vol. 14, pp. 77-88, 2004.

[17] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy,* vol. 32, pp. 1217-1241, 2003.

[18] S. Krishnamurthy, "Cave or community?," 2002.

[19] A. Bonaccorsi and C. Rossi, "Why open source software can succeed," *Research policy,* vol. 32, pp. 1243-1258, 2003.

[20] C. Subramaniam, R. Sen, and M. L. Nelson, "Determinants of open source software project success: A longitudinal study," *Decision Support Systems,* vol. 46, pp. 576-585, 2009.

[21] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy,* vol. 12, pp. 23-49, 1999.

[22] C. M. Schweik, R. C. English, M. Kitsing, and S. Haire, "Brooks' versus Linus' law: an empirical test of open source projects," in *Proceedings of the 2008 international conference on Digital government research*, 2008, pp. 423-424.

[23] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel," *Research policy,* vol. 32, pp. 1159-1177, 2003.

[24] S. Krishnamurthy, "On the intrinsic and extrinsic motivation of free/libre/open source (FLOSS) developers," *Knowledge, Technology & Policy,* vol. 18, pp. 17-39, 2006/12/01 2006.

[25] J. A. Roberts, I.-H. Hann, and S. A. Slaughter, "Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects," *Management science,* vol. 52, pp. 984-999, 2006.

[26] K. Crowston and B. Scozzi, "Open source software projects as virtual organisations: competency rallying for software development," 2002, pp. 3-17.

[27] Y. Fang and D. Neufeld, "Understanding sustained participation in open source software projects," *Journal of Management Information Systems,* vol. 25, pp. 9-50, 2009.

[28] C. Santos, G. Kuk, F. Kon, and J. Pearson, "The attraction of contributors in free and open source software projects," *The Journal of Strategic Information Systems,* vol. 22, pp. 26-45, 2013.

[29] J. Colazo and Y. Fang, "Impact of license choice on open source software development activity," *Journal of the American Society for Information Science and Technology,* vol. 60, pp. 997-1011, 2009.

[30] I. Qureshi and Y. Fang, "Socialization in open source software projects: A growth mixture modeling approach," *Organizational Research Methods,* vol. 14, pp. 208-238, 2011.

[31] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder," *Software, IEEE,* vol. 30, pp. 52-66, 2013.

[32] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng, "The impact of social media on software engineering practices and tools," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010, pp. 359-364.

[33] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, pp. 1277-1286.

[34] P. J. Guinan, J. G. Cooprider, and S. Faraj, "Enabling software development team performance during requirements definition: A behavioral versus technical

approach," *Information Systems Research,* vol. 9, pp. 101-125, 1998.
[35] P. Turney and M. L. Littman, "Unsupervised learning of semantic orientation from a hundred-billion-word corpus," 2002.
[36] S. R. Das and M. Y. Chen, "Yahoo! for Amazon: Sentiment extraction from small talk on the web," *Management Science,* vol. 53, pp. 1375-1388, 2007.
[37] N. Archak, A. Ghose, and P. G. Ipeirotis, "Deriving the pricing power of product features by mining consumer reviews," *Management Science,* vol. 57, pp. 1485-1509, 2011.
[38] A. Abbasi and H. Chen, "CyberGate: a design framework and system for text analysis of computer-mediated communication," *Mis Quarterly,* pp. 811-837, 2008.
[39] M. Chau and J. Xu, "Business intelligence in blogs: Understanding consumer interactions and communities," *MIS quarterly,* vol. 36, pp. 1189-1216, 2012.
[40] R. M. Cyert and J. G. March, "A behavioral theory of the firm," *Englewood Cliffs, NJ,* vol. 2, 1963.
[41] M. S. Feldman, "Organizational routines as a source of continuous change," *Organization science,* vol. 11, pp. 611-629, 2000.
[42] J. G. March and H. A. Simon, "Organizations," 1958.
[43] R. Nelson and S. Winter, *An evolutionary theory of economic change*. Cambridge: Belknap Press/Harvard University Press, 1982.
[44] M. S. Feldman and B. T. Pentland, "Reconceptualizing organizational routines as a source of flexibility and change," *Administrative Science Quarterly,* vol. 48, pp. 94-118, 2003.
[45] J. Gaskin, N. Berente, K. Lyytinen, and Y. Yoo, "Toward generalizable sociomaterial inquiry: a computational approach for zooming in and out of sociomaterial routines," *Mis Quarterly,* vol. 38, pp. 849-871, 2014.
[46] J. Gaskin, V. Thummadi, K. Lyytinen, and Y. Yoo, "Digital Technology and the variation in design routines: a sequence analysis of four design processes," 2011.
[47] J. E. GASKIN, D. M. SCHUTZ, N. Berente, and K. Lyytinen, "THE DNA OF DESIGN WORK: PHYSICAL AND DIGITAL MATERIALITY IN PROJECT-BASED DESIGN ORGANIZATIONS," in *Academy of Management Proceedings*, 2010, pp. 1-6.
[48] B. T. Pentland, T. Hærem, and D. Hillison, "The (N) ever-changing world: stability and change in organizational routines," *Organization Science,* vol. 22, pp. 1369-1383, 2011.
[49] A. Lindberg, N. Berente, J. Howison, and K. Lyytinen, "Variations in Information Processing Capacity: A Study of Routine Heterogeneity in Open Source Projects," presented at the Academy of Management Meeting, Vancouver, Canada, 2015.
[50] A. Lindberg, N. Berente, and K. Lyytinen, "Towards an Open Source Software Development Life Cycle: A Study of Routine Heterogeneity and Discourse across multiple Releases," in *Academy of Management Proceedings* Vancouver, Canada, 2015.
[51] S. E. Page, *Diversity and complexity*: Princeton University Press, 2010.
[52] D. T. Campbell, "Blind variation and selective retentions in creative thought as in other knowledge processes," *Psychological review,* vol. 67, p. 380, 1960.
[53] K. E. Weick and C. A. Kiesler, *The social psychology of organizing* vol. 2: Random House New York, 1979.
[54] B. Levitt and J. G. March, "Organizational learning," *Annual review of sociology,* pp. 319-340, 1988.
[55] A. Lindberg, "Understanding Change in Open Source Communities: A Co-evolutionary Framework," in *Academy of Management Proceedings*, 2013, p. 16619.
[56] W. N. Robinson and T. Deng, "Diversity in Software Development Routines are Attractive: A Preliminary Analysis of GitHub Repositories," in *Association for Information Systems Conference (AMCIS)*, Puerto Rico, 2015.
[57] D. Geiger and A. schröder, "EVER-CHANGING ROUTINES? TOWARD A REVISED UNDERSTANDING OF ORGANIZATIONAL ROUTINES BETWEEN RULE-FOLLOWING AND RULE-BREAKING," *Schmalenbach Business Review (SBR),* vol. 66, pp. 170-190, 2014.
[58] K. Conboy, "Agility from first principles: reconstructing the concept of agility in information systems development," *Information Systems Research,* vol. 20, pp. 329-354, 2009.
[59] S. Cant, D. R. Jeffery, and B. Henderson-Sellers, "A conceptual model of cognitive complexity of elements of the programming process," *Information and Software Technology,* vol. 37, pp. 351-362, 1995.
[60] J. E. Robbins and D. Redmiles, "Software architecture design from the perspective of human cognitive needs," in *Proceedings of the California Software Symposium (CSS'96)*, 1996, pp. 16-27.
[61] R. Subramanyam and M. S. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects," *Software Engineering, IEEE Transactions on,* vol. 29, pp. 297-310, 2003.
[62] S. W. Hansen, W. N. Robinson, and K. J. Lyytinen, "Computing Requirements: Cognitive Approaches to Distributed Requirements Engineering," in *System Science (HICSS), 2012 45th Hawaii International Conference on*, 2012, pp. 5224-5233.
[63] S. Hansen and K. Lyytinen, "Distributed Cognition in the Management of Design Requirements," 2009.
[64] B. Thummadi, K. Lyytinen, and S. Hansen, "Quality in Requirements Engineering (RE) Explained Using Distributed Cognition: A Case of Open Source Development," 2011.
[65] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE,* vol. 77, pp. 257-286, 1989.
[66] S. Kullback, *Information theory and statistics*: Dover Pubns, 1997.
[67] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word–emotion association lexicon," *Computational Intelligence,* vol. 29, pp. 436-465, 2013.
[68] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 5, pp. 3-55, 2001.
[69] A. Gabadinho, G. Ritschard, N. S. Mueller, and M. Studer, "Analyzing and visualizing state sequences in R with TraMineR," *Journal of Statistical Software,* vol. 40, pp. 1-37, 2011.