Empir Software Eng (2013) 18:375–431
DOI 10.1007/s10664-012-9205-0

Studying the impact of social interactions on software quality

Nicolas Bettenburg · Ahmed E. Hassan

Published online: 28 April 2012

Editor: Giulio Antoniol and Keith Gallagher

Abstract Correcting software defects accounts for a significant amount of resources in a software project. To make best use of testing efforts, researchers have studied statistical models to predict in which parts of a software system future defects are likely to occur. By studying the mathematical relations between predictor variables used in these models, researchers can form an increased understanding of the important connections between development activities and software quality. Predictor variables used in past top-performing models are largely based on source code-oriented metrics, such as lines of code or number of changes. However, source code is the end product of numerous interlaced and collaborative activities carried out by developers. Traces of such activities can be found in the various repositories used to manage development efforts. In this paper, we develop statistical models to study the impact of social interactions in a software project on software quality. These models use predictor variables based on social information mined from the issue tracking and version control repositories of two large open-source software projects. The results of our case studies demonstrate the impact of metrics from four different dimensions of social interaction on post-release defects. Our findings show that statistical models based on social information have a similar degree of explanatory power as traditional models. Furthermore, our results demonstrate that social information does not substitute, but rather augments traditional source code-based metrics used in defect prediction models.

Keywords Human factors · Software evolution · Metrics/measurement · Software quality assurance

N. Bettenburg (✉) · A. E. Hassan
Software Analysis and Intelligence Lab (SAIL), Queen's University, School of Computing, 156, Barrie Street, Kingston, ON K7L 3N6, Canada
e-mail: nicbet@cs.queensu.ca

A. E. Hassan
e-mail: ahmed@cs.queensu.ca

1 Introduction

In the foreword to "Why programs fail" (Zeller 2009), James Larus, director of Microsoft's Customer Care Framework (CCF) project notes: *"If software developers were angels, debugging would be unnecessary[...]"* as an homage to the famous words by James Madison. With this line, Larus expresses a fundamental software engineering problem that sparks enormous research efforts: software contains defects, and fixing these defects is very costly—even more so, if they are discovered after the software has shipped.

To reduce maintenance costs, researchers have extensively studied two core areas in empirical software engineering: understanding and minimizing the cause of defects and building effective systems to predict where defects are likely to occur in the software system (Bird et al. 2009). Both research areas are intertwined: knowledge gained from understanding root causes can help in building better predictors (Schröter et al. 2006), and at the same time the study of prediction models provides cues for understanding the causes of defects, such as complex code change processes (Hassan 2009). Past work in defect prediction makes extensive use of product and process metrics (Purao and Vaishnavi 2003), obtained from the source code of a software system, such as code complexity (McCabe 1976), code change metrics (Munson and Elbaum 1998), or inter-dependencies of elements in the code (Nagappan and Ball 2007).

However, source code is the end product of a variety of collaborative activities carried out by the developers of a software. Lately, researchers started to realize that the intricacies of these activities such as social networks (Wolf et al. 2009), work dependencies (Cataldo et al. 2009) and daily work routines (Śliwerski et al. 2005) impact the quality of a software product. Traces of these activities can be found in the repositories that developers use on a day to day basis, such as version archives, issue tracking systems, and email communication archives. In this study we investigate how we can use information about the social interactions in the community for defect modeling, and set out to study their relative impact on software quality. In particular, we focus on social information extracted from discussions on issues reports, which are stored in issue tracking systems. We use statistical models to establish and inspect mathematical dependencies between defects and social interaction metrics – an approach that has been successfully used in previous research to study the relation between source code metrics and defects (Cataldo et al. 2009; Hassan 2009; Nagappan and Ball 2005, 2007; Schröter et al. 2006; Zimmermann et al. 2007). In particular, we set out to study the following relations:

(1) The relationship between the social structures, extracted from discussions in issue tracking systems and software quality, as expressed through post-release defects.
(2) The relationship between the contents and characteristics of communication, measured through metrics computed from issue tracking system discussions, and software quality, as expressed through post-release defects.
(3) The relationship between workflow in the community, measured through activities in the issue tracking systems, and software quality, as expressed through post-release defects.

Through case studies on the ECLIPSE and Mozilla FIREFOX software systems, we find that such relationships exist and that they can be used to create prediction models with explanatory power similar to traditional models based on product and process metrics. In addition, we find that a combination of our model based on social interaction metrics and a traditional model based on product and process metrics, yields higher explanatory power than each of the models taken separately.

In particular, our work makes the following contributions to the research field. (1) We distill those metrics of social interaction, that are connected to software quality, and describe their effect through odds ratios. (2) We demonstrate that social information metrics complement traditional, source code based metrics, and investigate which of the social information metrics could be valuable for defect modelling.

The rest of this paper is organized as follows. Section 2 describes the set of social information metrics we use throughout this study. Section 3 presents the general design of our case studies, together with a discussion of the statistical (regression) models and the methods used to describe their performance.

Section 4 presents our case study on the ECLIPSE software system, and Section 5 presents our case study on the Mozilla FIREFOX software system. We discuss our findings of both case studies in more detail in Section 6, and perform a comparison of our observations across both projects. In Section 7 we investigate the possibility of combining models based on social information metrics with traditional models based on source code metrics. We close our work by discussing related research work (Section 8), possible threats to the validity of our study (Section 9), and our conclusions (Section 10).

This work extends our original study published at the 2010 International Conference on Software Comprehension (ICPC) as follows. (1) We have updated our analysis to better understand the impact of social interaction metrics on defects. (2) We have added a new case study on Mozilla FIREFOX. (3) We analyzed multiple releases of each software project, and (4) We have prepared publicly available datasets and scripts to help repeatability of our study (Appendix A).

2 Social Interaction Metrics

In this section we describe the four dimensions of social interaction metrics that we use in our statistical models. To help our readers follow along in the text and increase the readability of this work, we present an overview of these metrics in Table 1. For each metric, we briefly motivate its inclusion and outline our approach to measure it. Our social interaction metrics are determined from traces of activity that developers and users leave behind in the issue tracking system. Hence we calculate each metric on a per-issue report level.

As we will later study the relation between social interaction metrics and software quality on a per-file level, we need to aggregate values across all issue reports associated with a file. Our default method of aggregation is to take the *average*. However, for some metric we are interested in their variability and for these

Table 1 Reference of the measures of social interaction used in this study

Measure	Description
Baseline model	
CHURN	Code churn is defined as the number of lines added, modified, and deleted between two consecutive versions of a source code file.
Discussion contents	
NSOURCE	Number of source code regions found in a discussion by the infoZilla tool.
NSCOM	Average cyclomatic complexity of the code found in a discussion.
NPATCH	Number of patches found in a discussion by the infoZilla tool.
PATCHS	Number of files modified by a patch.
NTRACE	Number of stack traces found in a discussion by the infoZilla tool.
TRACES	Size of a stack traces in number of stack frames.
NLINK	Number of URL links to resources outside the discussion.
Social structures	
NPART	Number of unique participants in a discussion.
NUSERS	Number of unique participants in a discussion, who are users.
NDEVS	Number of unique participants in a discussion, who are developers.
CON1-3	Unique login names of the three most experienced developers participating in a discussion.
SNACENT	The degree to which each participant talks to other participants, captured through a measure of clonesness-centrality.
Communication dynamics	
NMSG	Total number of messages exchanged in a discussion.
DLEN	Number of words in a discussion.
DLENE	Variability in the number of words across all discussions on issue reports associated with a source code file.
REPLY	Mean reply time between the messages of a discussion.
REPLYE	Entropy of the mean reply time between discussions on issue reports associated with a source code file.
INT	Interestingness of an issue report, captured through the size of notification list.
INTE	Variability in the interestingness across all issue reports associated with a source code file.
Workflow	
WA	Workflow activity recorded for an issue report.
WAE	Variability in workflow activity across all issue reports associated with a source code file.

metrics we will use *entropy* for aggregation. Entropy is a concept we borrow from information theory (Shannon 2001). The normalized entropy is defined as:

$$H(P) = -\sum_{k=1}^{n} (p_k \cdot log_n(p_k)) \tag{1}$$

where $p_k \geq 0, \forall k \in 1, \ldots, n$ and $\sum_{k=1}^{n} p_k = 1$. Normalized entropy is an extension to Shannon's classical measure of entropy (Shannon 2001) and allows us to compare entropy metrics across different distributions. Measures of entropy have been used in previous research to study the evolution of code changes (Hassan 2009), noting that when a project is not managed well, or the code change process is not under control,

the system will be in a state of maximum entropy (chaos). Through measures of the entropy of the social processes surrounding code changes, we are studying whether this conjecture holds true within our context.

To illustrate normalized entropy, we consider the following example (presented in Fig. 1). Let m_A be a set of measures of the time (in hours) between the submission of consecutive discussion messages on issue A, and let m_B be a set of measures of the time (in hours) between the submission of consecutive discussion messages on issue B. Suppose we find that $m_A = \{1.1, 1.2, 1.3, 1.4\}$ and $m_B = \{1.1, 1.2, 4.0, 1.3\}$. Both sets of measures are the same, except that for one value: in m_B, the third measure turns out to be 4.0 hours between two messages. After normalization of measures, we obtain the sets of normalized values $\bar{m}_A = \{0.22, 0.24, 0.26, 0.28\}$ and $\bar{m}_B = \{0.14, 0.16, 0.17, 0.53\}$. We can now calculate the normalized entropy for both sets of measures, as presented in (1), and find that $H(\bar{m}_A) = 0.9971$ and $H(\bar{m}_B) = 0.8735$. In particular, we want to note the following: the set m_B contains a larger variability of measures, but exhibits a lower measure of normalized entropy. *In general, we achieve the maximum value of entropy, if all elements in the set have equal values; and we achieve minimum entropy, if all except one measure have a value of zero.*

2.1 Dimension One: Discussion Content

In this section, we describe the seven discussion content metrics that we use in our study. We choose to incorporate metrics on code examples, patches, stack traces and links found in discussions, to better understand the content of discussions and their impact. For example, Bird et al. note that technical talk (indicated by the presence of a larger amount of technical information items) can have a different impact than regular chitchat (Bird et al. 2008). We use the `infoZilla` tool (Bettenburg et al. 2008b) to extract technical information items from the textual contents of bug report discussions.

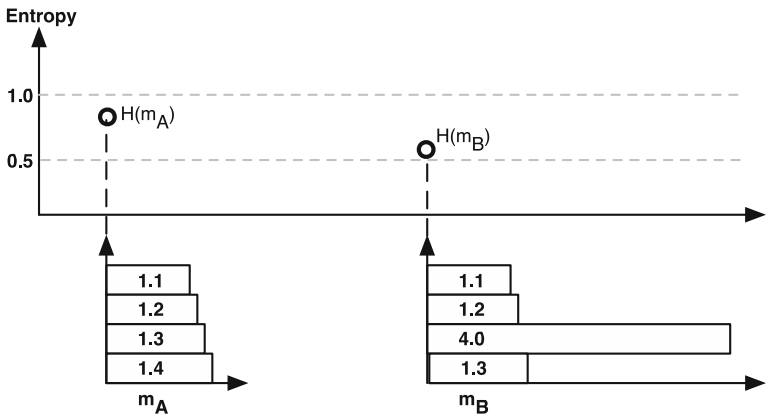

Fig. 1 Example of entropy: a larger variability in the data leads to a lower measure of normalized entropy

In a previous study (Bettenburg et al. 2008a), we asked developers of the ECLIPSE and MOZILLA projects, which of the information inside bug reports is most helpful for them when working on the reported issues. Among the top answers were information items like crash reports (in the form of stack traces), source code examples and patches. As a precise understanding of the underlying defect is crucial for addressing a reported issue adequately, we conjecture that the presence or absence of information items in bug reports can possibly influence the quality of the source code changes carried out under the context of the reported issue. For example, discussions of test cases can help developers to recover the rationale and intended correct behaviour of a software system.

2.1.1 Source Code Examples (Amount, Complexity)

Our first metric is the *amount of source code* (NSOURCE) present in a discussion. Source code can find its way into an issue report due to several reasons: reporters point out specific classes and functions they encountered a problem with, or provide smaller test-cases to exactly illustrate a misbehaviour; developers point users at locations in the source code they require more information about, and discuss possible ways to address an issue with peers. In particular, we are measuring the number of complete code examples, rather than lines of code, since source code often loses its original formatting when present in discussions. Our `infoZilla` tool reports source code examples, as the largest blocks of source code surrounded by either natural language text, or other structural elements. As the complexity of the code discussed might be an indicator for the intricacy of the reported problem and an indicator of future risk, we also compute the *complexity of the discussed code* (NSCOM) as a qualitative measure for each of the source code examples., we use McCabe's cyclomatic complexity (McCabe 1976), rather than lines of code as our complexity metric.

Our computation of McCabe complexity is analog to the implementation found in the open source static code checker PMD[1] and is defined as

$$McCabeComplexity = 1 + count(DecisionPoints, code) \tag{2}$$

with a decision point being either an `if`, `else`, `else if`, `for`, `while`, `do`, or `case` statement in the source code.

2.1.2 Patches (Amount, Filespread)

Our second metric is the *amount of patches* (NPATCH) provided in the discussions. Publicly discussed patches provide peer-reviewed solutions to the reported issues. Multiple patches can either present different solutions to the same problem, solutions to a variety of less complex subproblems, or be different revisions of the same solution that has been refined through the discussion. In addition to the amount of

[1] http://pmd.sourceforge.net/

patches we also record the *number of files changed by a patch* (PATCHS). Through this metric we capture the spread of a patch. We motivate this choice with the idea that patches resulting in large or wide-spread changes to the source code might negatively impact dependent parts of the code (even though they might correctly fix the reported issues) (Hassan 2009).

2.1.3 Stack Traces (Amount, Stacksize)

Our third metric records the *amount of stack traces* (NTRACE) provided in the contents. Information inside stack traces provides helpful information for developers to narrow down the source of a problem, and are hence valuable for finding and fixing the root causes of issues rather than addressing their symptoms (Zeller 2009). We use the number of methods reported in the stack traces as a measure for the *size of stack traces* (TRACES).

2.1.4 Links

Our fourth metric of discussion contents records the *amount of links* (NLINK) present. Developers and users use URLs to provide cross-references to related issues and to refer to external additional information that might be relevant to the original problem. We make no distinction between internal links (e.g., to other issue reports) and external links (e.g., to third-party websites).

2.2 Dimension Two: Social Structures

In addition to the information obtained from the textual contents of discussions, we compute a number of metrics to describe the social structures between developers and users, created through issue report discussions. In the following we describe the five metrics of social structures used in our study.

2.2.1 Discussion Participants

In order to contribute to the BUGZILLA system, users have to sign in with a username and password. The username acts as a unique handle for all activity in the issue tracking system. We conjecture that the total amount of unique participants in the discussion of an issue report is an indicator of the relative importance of the reported problem. Our first measure hence counts the *number of unique participants* (NPART) in the discussion.

2.2.2 Role

In this study we further categorize participants into two different roles: developers and users. We consider a participant to be a developer, if he was assigned to fixing at least one BUGZILLA issue in the past. By measuring the *number of unique users* (NUSERS) and the *number of unique developers* (NDEVS) participating in the discussions we can distinguish between internal discussions (more developers than users), external discussions (more users than developers) and balanced discussions (even amount of developers and users).

2.2.3 Experience

Another social property of participants, orthogonal to their role, is their degree of experience in the community. In our study, we determine the top three participants with the highest experience (expressed by the past amount of contributed messages) for each discussion attached to an issue report. These measures are captured in the three variables (CON1), (CON2), and (CON3). In particular, each variable CON1, CON2, and CON3 contains the unique Bugzilla login names of the three developers, we determined to be the most experienced. The degree of experience of a participant can influence the development process connected to an issue; for example Guo et al. show that defects reported by more experienced users have a higher likelihood to get fixed (Guo et al. 2010).

2.2.4 Centrality

Our last metric of social structure (SNACENT) is taken from the area of social network analysis, called *closeness-centrality*. This metric is commonly used in social network analysis to describe the efficiency of spreading information among a group of people (Wasserman and Faust 1994). We conjecture, that inefficient relay of crucial information might have a negative impact on software quality. We measure closeness-centrality as follows. For each discussion attached to an issue report in the bug database we first construct a discussion flow graph (Mertsalov et al. 2009). The discussion flow graph is an undirected graph that has participants as nodes and contains an edge for every pair of two consecutive messages in the discussion, connecting both message senders. We express the interconnectedness of nodes (participants) in the discussion flow graph as a measure of Wasserman and Faust (1994).

The closeness-centrality C_C of each participant in a discussion is the inverse of the average shortest-path distance d_S from the participant in the discussion flow graph to every other participant in the graph. Figure 2 illustrates an example discussion and corresponding discussion flow graph. In this example, participant A is connected to participants B and D directly. Participant C is connected only to participant B,

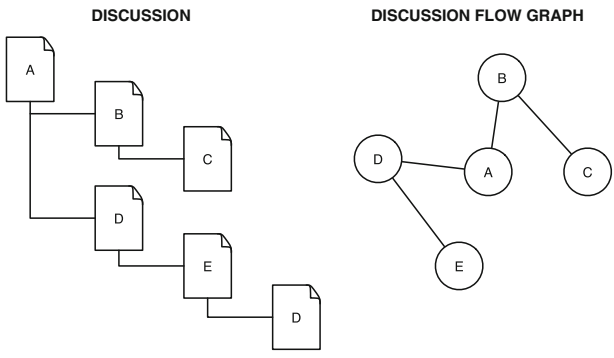

Fig. 2 Example discussion and resulting discussion flow graph

and participant E is connected only to participant D. The closeness-centrality for participant E is

$$C_C(E) = \left(\frac{1}{4} \cdot (d_S(E, A) + d_S(E, B) + d_S(E, C) + d_S(E, D))\right)^{-1}$$
$$= \left(\frac{1}{4} \cdot (2 + 3 + 4 + 1)\right)^{-1}$$
$$= \frac{2}{5}$$

Similarly, the closeness-centrality value for participant A is $\frac{2}{3}$, and for both participants B and D it is $\frac{4}{7}$. Since closeness-centrality is a per-node measure (one value for each discussion participant, and one such set of values for each discussion associated with a particular file), we aggregate the closeness-centrality of all participants in the discussion into a single value, through normalized entropy. The more participants are equally able to spread information to everybody else, the higher the normalized entropy measure will be.

2.3 Dimension Three: Communication Dynamics

In addition to information about discussion content and involved participants, we attempt to measure discussion activity both quantitatively and qualitatively. In this context, we refer to communication dynamics as the changing attributes of a discussion as it unfolds when new discussion activity is added. In the following we describe the six metrics of communication dynamics, used in our study.

2.3.1 Number of Messages

By their very nature, issues that are complex, not well understood, or controversial require a greater amount of information exchange relative to simple issues. We capture this intuition through a measure of the *amount of messages* (NMSG) exchanged in a discussion.

2.3.2 Length of Messages

Following the same intuition, we define two additional metrics: first, the *number of words in a discussion* (DLEN), and second *discussion length entropy* (DLENE), as a proxy to the variability in wordiness in discussions. We consider the "wordiness" of messages as an indicator for the cognitive complexity of the reported issue and greater fluctuations of wordiness (resulting in a higher measure of entropy) as an indicator for possible communication problems.

2.3.3 Reply Time

Cognitive science defines communication as "the sharing of meaning" (Alatis 1993; D'Este 2004). The absence of communication for an extended period of time, or distorted communication, is related to misinterpretations and misunderstandings.

In the context of software development, such misinterpretations when carrying out changes to the source code can be the cause of errors. We capture this idea by measuring both, the *mean reply time between messages* (REPLY), and the *reply time entropy* as a proxy to the variability in reply times (REPLYE) in discussions. We conjecture that a higher variation in reply times (e.g., a long pause in an otherwise fast-paced discussion) captures temporal anomalies in the discussion flow that might indicate potential problems, and thus possibly post-release defects.

2.3.4 Interestingness

The BUGZILLA system allows users to get automatic notifications when an issue report is changed, via so-called "CC-lists". We use a measure of the number of people who signed up for such notifications as an indicator of the *interestingness* (INT) of an issue report. This measure is different from the *number of participants*, since users of the issue tracking system can be on the notification list, while not contributing to the discussion for an issue report. In addition we capture the variability of interestingness in a measure of *interestingness entropy* (INTE). We conjecture that a larger variability of interestingness (e.g., a rather uninteresting file suddenly becomes very interesting, versus a file that is always rather uninteresting), might thus be an potential indicator for post-release defects.

2.4 Dimension Four: Workflow

Issue reports represent work items for developers and follow a set of states from creation until closure (Anvik et al. 2006) and transitions between these states create a workflow. Any workflow activity associated with each report is recorded in the BUGZILLA system. We conjecture that a high workflow activity indicates anomalies (Guo et al. 2011; Jeong et al. 2009; Shihab et al. 2010a), such as re-assignment of the work item to another developer, or re-opening reports that were previously marked as completed. To capture workflow activities, we measure the *total amount of workflow activity* (WA) associated with each issue report, as well as the *entropy of workflow activity* (WAE) to capture variability in the process.

3 Study Design

Our analysis uses statistical models to investigate the relations between social information and software quality. We do so by exploring the statistical relations between the failure proneness of files and the social information metrics captured from issue reports associated with these files. In this section we describe the design of our study, our model-building process and the results of our comparative analysis between the model based on social information metrics and classical models using code-based product and process metrics. Following previous work in defect prediction (Nagappan and Ball 2007), we divide the collection of measurements into two distinct phases. For a period of 6 months before a release of the software we capture the social interaction metrics described in Section 2 for each file that has at least one issue report associated with it. We then measure the amount of defects (POST) reported for each file for the next 6 months following the release. For both case studies, we choose to perform our measures for periods of 6 months surrounding the releases

of `Eclipse 3.0`, `Eclipse 3.1`, and `Eclipse 3.2`, as well as Mozilla `FIREFOX 1.5`, Mozilla `FIREFOX 2.0`, and Mozilla `FIREFOX 3.0`. From the measurements obtained for the corresponding time periods, we create regression models that set the occurence of *post-release defects* into relation of our *pre-release measures*. The complete regression model has the following form:

$$\begin{aligned} Prob\,(Postrelease \quad Defects) = \theta \cdot CodeChurn \\ + \sum_i \alpha_i \cdot DiscussionContentsMetric_i \\ + \sum_j \beta_j \cdot SocialStructuresMetric_j \\ + \sum_k \gamma_k \cdot CommunicationDynamicsMetric_k \\ + \sum_l \delta_l \cdot WorkflowMetric_l + \epsilon \end{aligned} \tag{3}$$

Here, ϵ is called the *intercept* of the model, and $\theta, \alpha_i, \beta_j, \gamma_k$, and δ_l are called the *regression coefficients*. Based on this model, we will investigate the statistical relationships between the social interaction metrics of each dimension, which are used as the *regression variables* in the model, and the probability of post release defects, which is used as the *dependent variable* in the model. We start with a preliminary analysis of the regression variables using descriptive statistics, to illustrate general properties of the collected metrics. Next, we perform a correlation analysis to consider possible inter-relations between measurements. We then construct several logistic regression models to investigate the relative impact of each of the four dimensions of social interactions metrics on the risk of post-release defects. Our approach is similar to the work by Cataldo et al. (2009) and Mockus et al. (2005).

We follow a hierarchical modelling approach when creating all our models: we start out with a baseline model that uses code churn, a classical defect predictor, as the regression variable. We then build subsequent models to which we step-by-step add our content, structure, communication dynamics and workflow metrics, and report for each model the explanatory power, χ^2, of the model. The χ^2 statistic can be thought of as a measure of "goodness of contribution from the set of regression variables" (Cohen 2003). In addition, we report for each model M_i the percentage of deviance explained by the model, which is a quality of fit statistic and is defined as

$$D(M_i) = -2 \cdot LL(M_i) \tag{4}$$

with $LL(M_i)$ denoting the log-likelihood of the model, and the deviance explained as a ratio between $D(M_0) = D(Defects \sim Intercept)$ and $D(M_i)$. This statistic is similar to the coefficient of determination, R^2, and describes the variability in the data set the model accounts for (Steel and Torrie 1960), and thus describes how well the model, when used for prediction, describes future outcomes. Our choice of χ^2 over R^2 as a measure of goodness is rooted in two observations. First, in logistic regression models, R^2 needs to be approximated through a pseudo-R^2 measure, whereas χ^2 is directly accessible. Second, as we add more regression variables to our logistic

regression models through the hierarchical approach taken, the pseudo-R^2 measure increases, even if the added variables add no value to the regression models.

Overall, a hierarchical modelling approach has the advantage over a step-wise modelling approach that it minimizes artificial inflation of errors, and thus over-fitting (Cataldo et al. 2009). To determine the contribution of each dimension of social metrics to our model, we test for each subsequent model whether the difference in explanatory power from the previous model is statistically significant, and present the corresponding p-level.

To ease readability and interpretability of our results, we present the odds ratios (Edwards 1963) of each regression coefficient, rather than the raw β-coefficients. For a presentation of the β-coefficients of each model, we refer our reader to Appendix B. Logistic regression models express regression coefficients as the log of odds, and the relationship between odds ratios (OR) and regression coefficients (β) is expressed through:

$$OR_i = e^{\beta_i} \qquad and \qquad \beta_i = log(OR_i)$$

An odds ratio greater than one indicates a positive relation between the dependent variable (risk of post-release defects) and the independent variables (social interaction metrics), whereas an odds ratio between zero and one indicates a negative relation. As we are working in a log-transformed space, the odds-ratios have to be interpreted accordingly: a single unit change in the log-transformed space corresponds to a change from 1 to 2.71 ($= e^1$) units in the untransformed space.

As a word of warning we want to note that odds ratios should not be compared directly. Odds ratios describe the effect of a one-unit increase of the regression variable on post-release defects, while keeping every other regression variable constant. However, since regression variables attain different actual values in both projects their relative effects might be different across projects. As an example consider the following imaginary regression variable X. If we would find that X has an odds ratio of 1.5 in ECLIPSE, but an odds ratio of 2.0 in FIREFOX, one might be tempted to argue that X has a larger effect on defects in FIREFOX than in ECLIPSE. Yet, it is possible that X only attains actual values between 0.1 and 0.5 in FIREFOX, whereas it could attain actual values between 1.0 and 2.0 in ECLIPSE, thus the true effect of X on defects is considerably larger for ECLIPSE than FIREFOX—despite the smaller odds ratio!

4 Case Study One: Eclipse IDE

4.1 Data Collection

For our case study on the ECLIPSE project, we used two main sources of available data for ECLIPSE. First, we obtained a copy of the project's BUGZILLA database. This database collects modification requests that are submitted electronically by a reporter. These requests are commonly referred to as "bug reports". However, we find this term misleading as not all reported issues are defects (Antoniol et al. 2008) and for the remainder of this paper we will refer to bug reports as "issue reports". Every report contains a variety of supporting meta-information such as a unique identification number, the software version, operating system, or the reporter's

perceived importance. In addition, entries contain a short one-line summary of the issue at hand, followed by a more elaborate description. After submission, entries are discussed in more detail between developers and users, who provide further comments. In our data, we treat the initial description written by the reporter as the first message, starting a discussion. Overall, we collected a total of 300,000 issues submitted to the BUGZILLA system between October 2001 and January 2010.

The second data source we use is the source code archive of the ECLIPSE project. We obtained a snapshot of the CVS software repository, which contains the project's source code, as well as all the information about past changes that have been carried out by developers. To record which files were changed together in the form of a transaction, we perform a grouping of single change records using a sliding window approach (Śliwerski et al. 2005). Overall, we collected 977,716 changes (accounting for 224,643 transactions) carried out between October 2001 and December 2009.

In order to link information from both repositories together, we automatically inspect the transaction messages to identify pointers to issue reports. Each number mentioned in a transaction message is treated as a potential link to an entry in the bug database. Our algorithm initially assigns low trust to each potential link, but this trust increases when we find additional clues of the link's validity, such as keywords like "bug" or "fix", or common patterns used to mark references like "#" followed by a number. This approach was used in previous research (Fischer et al. 2003; Schröter et al. 2006; Śliwerski et al. 2005; Čubranić and Murphy 2003) with high success. To further increase the quality of links, we incorporated the improvements by Bird et al. (2009). Through these links we can then associate issue reports with files. Overall, we were able to establish 67,705 such links.

We used these links to compute social interaction metrics for the 6 months periods before the three major releases of ECLIPSE 3.0 (released June 21st, 2004), ECLIPSE 3.1 (released June 28th, 2005), and ECLIPSE 3.2 (released June 30th, 2006), and to count the amount of post-release defects for the 6 month periods after each release. To help our reader following along with our case study, we first present a detailed analysis and interpretation of our statistical models for ECLIPSE 3.0, followed by a discussion and comparison of models for releases 3.1 and 3.2 at the end of the section.

4.2 Preliminary Analysis of Social Interaction Measures for ECLIPSE 3.0

Our four dimensions of social interaction metrics (content, structure, dynamics, and workflow) represent different characteristics of collaborative activity on issues. While content metrics are more explicit in capturing the information exchanged between developers and users, our metrics of social structures are more implicit and capture the latent relationships and roles of stakeholders. Table 2 presents a summary of our metrics in the form of descriptive statistics.

Due to a relatively high amount of skew, we apply a standard log transformation to each social interaction measurement to even out the skewing effects during modelling (Bland and Altman 1996). Figure 3 summarizes the pairwise correlations between our 20 regression variables and our dependent variable in a correlogram visualization (Friendly 2002). A correlogram reports for each unique pair of variables the strength of the correlation as a colour-coded field (red for positive

Table 2 Descriptive statistics of social interaction measures for ECLIPSE 3.0

	Mean	SD	Min	Max	Skew
POST	1.16	2.28	0.00	35.00	5.00
NSOURCE	0.86	2.48	0.00	48.00	7.14
NSCOM	0.27	0.49	0.00	5.00	2.77
NPATCH	0.02	0.24	0.00	5.00	17.17
PATCHS	0.01	0.11	0.00	3.00	13.26
NTRACE	0.14	0.44	0.00	9.00	7.82
TRACES	3.56	10.73	0.00	175.00	5.04
NLINK	0.20	0.91	0.00	8.00	7.02
NPART	3.61	3.89	1.00	40.00	7.48
NDEVS	2.94	1.46	1.00	12.00	2.78
NUSERS	0.67	2.81	0.00	28.00	8.44
SNACENT	0.19	0.07	0.00	0.51	0.43
NMSG	7.32	5.92	2.00	67.00	3.13
REPLY	122.32	206.99	0.00	3239.00	5.17
REPLYE	0.10	0.09	0.00	1.00	1.29
DLEN	337.00	441.75	2.00	6259.00	4.60
DLENE	0.23	0.10	0.00	1.00	0.08
INT	3.80	8.42	0.00	55.00	4.94
INTE	0.14	0.26	0.00	1.00	1.74
WA	9.33	6.36	0.00	49.00	1.68
WAE	0.17	0.19	0.00	1.00	0.65

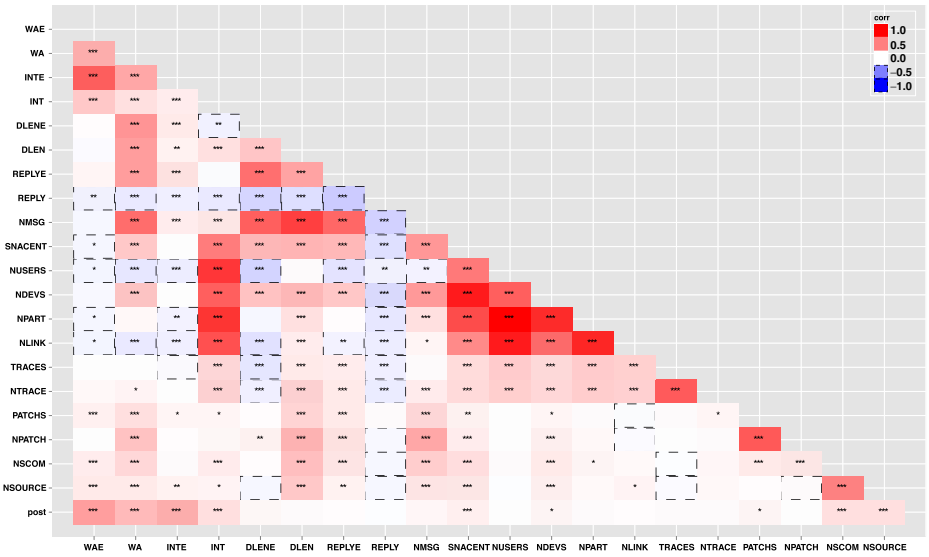

Fig. 3 Pairwise correlations of social interaction metrics in ECLIPSE 3.0. The strength of correlations is indicated by *fill* intensities; negative correlations are marked with a *dashed* outline. The row labeled "post" refers to post-release defects. *Stars* denote the p-level as follows: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

correlation, blue for negative correlation) and the p-level at which the correlation is significant. This visualization technique allows us to identify "hotspots" that need our attention.

We identify a number of intercorrelations between metrics from different dimensions in our dataset that could pose problems in our statistical modelling. For example, the measure of interestingness (INT) has a moderate to high correlation with our measures for number of users (NUSERS), number of participants (NPART), number of developers (NDEV), and number of links (NLINK). These correlations are statistically significant at $p < 0.001$. The first correlation between interestingness (INT) and the number of participants (NPART) stems from a default setting in the ECLIPSE issue tracking system, which puts contributors automatically on a notification list for any updates to the issue. Some of the observed correlations can be explained by a certain inherent amount of redundancy in the collected data. For example, the number of participants (NPART) is highly correlated with the number of users (NUSERS) and number of developers (NDEVS). However, our motivation for incorporating such redundancy is to investigate whether splitting up the information into more specialized representations helps to improve our model. The same intuition holds for the measure of centrality (SNACENT).

Since we observe a substantial number of high correlations among regression variables, we have to examine potential issues due to multi-collinearity among the variables. Even though the reliability of the statistical model as a whole is not affected by multi-collinearity issues, strong correlations among independent variables often leads to an error in estimates of the regression coefficients that we later use to investigate the relative impact of each variable on post-release defects. One widely adopted approach to reducing multi-collinearity is Principal Component Analysis (PCA). However, in the context of our research goal, PCA is a poor choice due to a disadvantageous side-effect of the approach (Shihab et al. 2010b). The result of PCA is a new set of regression variables, or principal components, which are linear combinations of all the input variables. As a result, we can not analyze the effects of the original regression variables anymore, which is the very purpose of this study. As an alternative way to address the problem of multi-collinearity, we perform a stepwise refinement of the set of variables we use through measuring the variance inflation factors for each variable. Variance Inflation Factors (VIF) are widely used to measure the degree of multi-collinearity between variables in regression models (Kutner et al. 2004).

Since the cut-off value for variance inflation factor analysis is a heavily disputed topic throughout statistical literature, we decided to follow the example by Kutner et al. 2004, and remove those variables from the model that have a variance inflation factor greater than 10. We start our analysis with a regression model that contains all our variables. The VIFs for this model are presented in Table 3, Model 1. We observe two variables that have a VIF greater than 10. We remove the highest one (NMSG) from the regression model and recompute the VIFs with the reduced set of variables. The resulting model, (Model 2 in Table 3) contains only one more variable with a VIF larger than 10. We remove the regression variable (SNACENT) from the model and recompute the inflation factors. In the resulting model (Model 3 in Table 3), no variables have a VIF larger than 5 and we finish our analysis of multicollinearity.

Table 3 Step-wise analysis of multicollinearity in the ECLIPSE 3.0 dataset

$\log(Y_i)$	Variance inflation factor		
	Model 1	Model 2	Model 3
NSOURCE	3.38	3.38	3.40
NSCOM	3.34	3.34	3.36
NPATCH	3.94	3.88	3.90
PATCHS	3.84	3.82	3.84
NTRACE	4.62	4.60	4.57
TRACES	4.78	4.75	4.70
NLINK	2.24	2.22	1.90
NDEVS	9.32	9.27	1.91
NUSERS	4.55	4.54	2.30
SNACENT	10.66	**10.65**	—
NMSG	**11.63**	—	—
REPLY	1.17	1.17	1.17
REPLYE	2.04	1.91	1.90
DLEN	4.21	1.91	1.87
DLENE	4.65	1.98	1.96
INT	2.82	2.82	2.60
INTE	1.71	1.71	1.71
WA	2.26	1.99	1.96
WAE	2.08	2.06	2.02

Bold entries denote variables that have been selected for removal during VIF analysis (highest VIF measures in each model)

4.3 Hierarchical Analysis

After having determined the reduced set of regression variables with low multicollinearity, we proceed by investigating the relative impact of each of the four dimensions of social interaction metrics on the post-release defects.

The results of our hierarchical analysis are presented in Table 4. We start our hierarchical analysis with a *baseline model* which relates code churn (number of lines added, deleted, or modified in a file from one version to another) (Munson and Elbaum 1998) to post-release defects. Code churn has been shown in the past to be one of the best code-based predictors of defects (Nagappan and Ball 2005, 2007), even when used across projects (Zimmermann et al. 2009). We obtained a measure of churn by mining the change histories of each file in the project's version control system. The results for the baseline model are presented in column MB of Table 4 and show that *CHURN* is positively associated to the failure proneness of a file during the post-release period. As expected, these results are in line with earlier findings (Nagappan and Ball 2005, 2007).

Model M1 introduces the first dimension of social interaction: metrics concerned with the contents of issue report discussions. The results of the logistic regression model show that only specific structural information items are statistically significant. When looking at the odds ratios, we observe a significant relationship between the number of files modified by patches (PATCHS) and future failure proneness of files. This result confirms earlier findings on the risk of scattered changes (Hassan 2009).

The second observation we make is a positive link between the number of source code samples (NSOURCE) and future defects. This is surprising, as we initially expected code samples to have a beneficial effect (as explained earlier in the motivation of this metric). One possible explanation might be that developers trust user provided sample solutions and incorporate their proposed (yet possibly flawed)

Table 4 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics in ECLIPSE 3.0

log(Y_i)	MB		M1		M2		M3		M4		M5	
CHURN	**4.996**	⋆	**4.631**	⋆	**4.658**	⋆	**5.303**	⋆	**3.688**	⋆	**4.470**	⋆
NSOURCE			**1.694**	⋆	**1.698**	⋆	**1.772**	⋆	**1.769**	⋆	**1.667**	⋆
NTRACE			0.79		0.768		0.864		0.881		1.115	
NPATCH			**0.209**	∘	**0.210**	∘	**0.284**	+	**0.231**	∘	0.291	
NSCOM			1.218		1.194		1.246		1.208		1.244	
PATCHS			**12.607**	∘	**12.626**	∘	**11.200**	∘	**12.736**	∘	**18.207**	•
TRACES			1.016		1.012		1.004		0.989		0.975	
NLINK			**1.764**	⋆	**1.613**	•	**1.600**	•	**1.666**	•	**1.596**	+
NPART					2.481		2.888		4.480		4.542	
NDEVS					0.475		0.582		0.385		0.274	
NUSERS					0.749		0.803		0.692		0.792	
REPLY							1.019		0.986		0.982	
REPLYE							**0.117**	⋆	**0.082**	⋆	**0.044**	⋆
DLEN							0.936		**0.898**	∘	**0.876**	+
DLENE							2.499		1.251		2.044	
INT							**0.829**	•	**0.821**	•	0.963	
INTE							1.109		1.013		1.306	
WA									**1.432**	⋆	**1.224**	+
WAE									**2.718**	∘	2.169	
CON1-3											**Fig. 4**	⋆
χ^2	**559.01**	⋆	**698.5**	⋆	700.15		**731.5**	⋆	**752.3**	⋆	**1055.19**	⋆
Dev. Expl.	10.71%		13.38%		13.41%		14.02%		14.41%		26.07%	
$\Delta\chi^2$			139.48		1.652		31.357		20.28		302.87	

For every stepwise model, we report odds ratios for each regression coefficient, as well as a goodness of fit metric (χ^2), the percentage of variation in the data each model explains (Dev. Expl) and the difference in goodness of fit compared to the previous hierarchical modelling step ($\Delta\chi^2$)

Bold entries denote statistically significant coefficients with a p level lower than 0.05

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

modifications without further verification; another explanation might be that code, which warrants an extended discussion, is more complex and thus more error-prone.

Furthermore, we observe a strong relationship between the number of links (NLINK) provided by users and failure proneness. One possible explanation for this relationship could be, that links to additional information (such as related issues) act as an indicator for hard-to-fix, or complex problems.

Summary: *Overall the results show that discussion content metrics are indicators of increased future failure proneness of a file. The explanatory power of the model increases by 2.67% over the baseline model and this increase is statistically significant.*

Model M2 introduces the second dimension of social interaction metrics: information on social structures. The results show that the role of participants and the overall amount of participants in a discussion have no statistically significant impact on the future failure proneness of files. As a result we see no increase in the explanatory power of the model by introducing the role of participants. We left out the measures of experience from this model, as we record them as factors with many levels that may

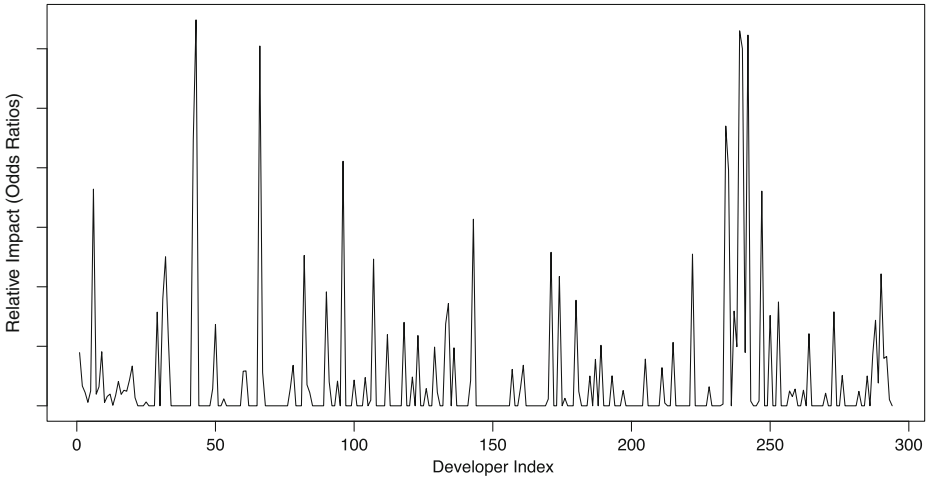

Fig. 4 Odds ratios for experience metrics in model M5, each index represents one distinct level (developer) of the factor variables

disrupt our hierarchical modelling approach. We will revisit these measures later in model M5.

Summary: *Overall, we cannot find a significant relation between the role of participants and post-release defects. The explanatory power of the extended model increases only marginally, however this increase is not statistically significant.*

Model M3 introduces metrics from the category of communication dynamics. The results show a statistically significant and strongly negative relation between the measure of reply time entropy and future failure likelihood, yet there exists no statistically significant relation between the related quantitative measure of average reply time length. The link between reply time entropy and failure proneness stays strong throughout the hierarchical process and indicates a relevant relationship. The second relation that we find is a moderately negative relation between the interestingness of an issue report and post-release defects. This relation however becomes irrelevant at a later point, when we introduce experience in model M5.

Summary: *Overall, we observe a strong effect of discussion flow inconsistencies on the failure proneness of files associated with the discussion. Even though the explanatory power of the extended model increases by only 0.61%, this increase is statistically significant.*

Model M4 introduces the last category of social interaction metrics used in our study: workflow activity. Our results show a positive relationship (with respect to the odds ratios of the corresponding regression coefficients) between the total amount of workflow activities and post-release defects. In particular, increased workflow activity and workflow activity entropy are both connected to a substantial increase in post-release defects.

Summary: *Our findings suggest that workflow activities play a marginal, complementary role in the relation between social interaction metrics and post-release defects. This is also indicated by the minor, yet statistically significant increase of explanatory power of the extended model (0.39%) when adding workflow activity metrics.*

We revisit the dimension of social structures in Model M5, by adding our measures of experience. These measures are expressed as three factors (with the unique login handles of discussion participants as levels) and record the three most experienced contributors for each discussion. When used in a regression model, each factor generates a large quantity of binary variables (as many as it has different levels). As a consequence, we do not show the complete model containing all these binary variables. However, we measure the inherent effect of the factors on the statistical model, using a random-effect analysis of variance (often referred to as type II ANOVA test), and present a plot of the odds ratios of each factor level in Fig. 4. As contributors are uniquely identified in the ECLIPSE issue tracking system by their email addresses, we do not include their names in this paper for privacy reasons; instead we anonymized each name by assigning a unique number. Our analysis of variance tests for the experience measures shows that they are statistically significant at $p < 0.001$. From the plot of odds ratios for each developer in Fig. 4, we observe that certain experienced participants in a discussion are strongly related to the presence of post-release defects (indicated by the spikes of the relative odds ratios in the plot).

The increase in explanatory power of model M5 is over-proportionally large (compared to the effect of the previous four dimensions). As a result, we performed further analysis to determine, whether the inclusion of the experience metric leads to overfitting in the statistical model (i.e., the effect captured in this metric describes random observations, rather than a real underlying relationship).

To judge possible overfitting, we divide our complete set of data into a training set (90% of the data) and a testing set (10%) of the data. On both sets we train a Model M5 and compare the χ^2 values. We repeat this process ten times, with random 90/10 splits (often referred to as "10-fold cross validation"). In each of these 10 runs, we observe a large difference (corresponding to differences between 10.12% and 16.31% of deviance explained) in χ^2 values between the model obtained from the training set and the model obtained from the testing step. These large differences confirm that CON1-CON3 indeed lead to an overfitting of the regression model and should thus not be included in the statistical model(s).

Summary: *Overall, the experience metric increases the explanatory power of the extended model significantly. The increase of 11.66% is statistically significant. The plot of odds ratios for each developer determined as experienced suggests that particular contributors in a discussion act as an indicator for future failure proneness. However, we found that the inclusion of this metric leads to a severe over-fitting in the model. As a result, we need to exclude CON1-CON3 for the rest of this paper, and perform any comparisons between projects and releases based on model M4.*

4.4 Additional Versions of ECLIPSE

We repeated the same hierarchical modelling approach for two additional releases of ECLIPSE. The final models are presented in Tables 5 and 6. During

multi-collinearity analysis, we removed NDEV and NMSG in the case of ECLIPSE 3.1, and NPATCH, SNACENT, as well as NMSG in the case of ECLIPSE 3.2. As a result, the SNACENT regression variable was only available for building models of ECLIPSE 3.1, in which it was not deemed statistically significant. Overall, we observe a similar increase of explanatory power as in our detailed case study of ECLIPSE 3.0. Each introduction of the four dimensions of metrics adds a statistical significant (yet relatively small) amount of information to the models. Notably, in ECLIPSE 3.1 and ECLIPSE 3.2, the information added by the second dimension (social structures), is deemed statistically significant, as opposed to ECLIPSE 3.0.

Overall, we find the following regression variables consistent across all studied releases: code churn (CHURN), number of source code examples (NSOURCE), number of files modified by patches (PATCHS), number of stack traces (NTRACE), discussion length (DLEN), and variability of interestingness (INTE). Among these metrics, CHURN, NSOURCE, PATCHS, and INTE are associated with an increased risk of post-release defects, whereas NTRACE and DLEN are associated with a reduced risk of post-release defects. Our findings confirm previous work on the relationship between code churn and defects (Nagappan and Ball 2005), modification spread and defects (Hassan 2009), as well as the helpfulness of stack traces when correcting defects (Schroter et al. 2010). The observed relationship

Table 5 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for ECLIPSE 3.1

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**6.573**	⋆	**5.723**	⋆	**5.668**	⋆	**5.253**	⋆	**3.295**	⋆	**3.873**	⋆
NSOURCE			**1.340**	⋆	**1.239**	•	**1.338**	⋆	**1.314**	⋆	**1.378**	⋆
NSCOM			**0.613**	⋆	**0.597**	⋆	**0.612**	⋆	**0.586**	⋆	**0.651**	•
PATCHS			1.552		1.709		1.678		1.799		2.612	
NPATCH			1.259		1.050		1.601		1.450		1.817	
NTRACE			0.701	+	**0.643**	∘	**0.668**	∘	0.730		0.778	
TRACES			1.094	+	**1.123**	∘	**1.157**	•	1.090	+	1.100	+
NLINK			**0.274**	⋆	**0.270**	⋆	**0.252**	⋆	**0.240**	⋆	**0.352**	⋆
NUSERS					**3.480**	⋆	**4.240**	⋆	**5.040**	⋆	**3.887**	⋆
SNACENT					**0.254**	⋆	**0.143**	⋆	**0.212**	•	**0.246**	∘
REPLY							0.972	+	**0.917**	⋆	**0.958**	∘
REPLYE							**3.610**	⋆	**2.565**	∘	**2.514**	∘
DLEN							**0.761**	⋆	**0.717**	⋆	**0.753**	⋆
DLENE							**27.210**	⋆	**6.561**	⋆	**3.018**	∘
INT							1.081	+	1.071		**1.149**	∘
INTE							**1.545**	•	1.144		1.376	+
WA									**1.710**	⋆	**1.410**	⋆
WAE									**3.924**	⋆	**3.618**	⋆
Chi Sq.	1643.98		**2134.46**	⋆	**2434.56**	⋆	**2644.3**	⋆	**2744.02**	⋆	**4288.3**	⋆
Dev. Expl.	11.66%		15.15%		17.27%		18.75%		19.46%		30.41%	
Delta Chisq			490.48		300.1		209.74		99.72		1544.28	

For every stepwise model, we report odds ratios for each regression coefficient, as well as a goodness of fit metric (χ^2), the percentage of variation in the data each model explains (Dev. Expl) and the difference in goodness of fit compared to the previous hierarchical modelling step ($\Delta\chi^2$)

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

Table 6 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for ECLIPSE 3.2

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**4.614**	⋆	**4.277**	⋆	**4.312**	⋆	**3.332**	⋆	**4.867**	⋆	**5.809**	⋆
NSOURCE			**1.991**	⋆	**1.933**	⋆	**2.042**	⋆	**2.056**	⋆	**1.894**	⋆
NSCOM			**0.572**	⋆	**0.604**	⋆	**0.564**	⋆	**0.552**	⋆	**0.487**	⋆
PATCHS			1.592		1.525		1.648		1.791		2.083	
NTRACE			**0.581**	∘	**0.566**	∘	0.690		0.678		0.604	
TRACES			**1.223**	⋆	**1.225**	⋆	**1.167**	∘	**1.182**	•	1.133	+
NLINK			**1.254**	⋆	**1.223**	•	1.136		1.056		**1.543**	⋆
NDEVS					**0.882**	∘	**0.814**	•	**0.785**	⋆	0.869	
NUSERS					**1.186**	•	1.126	+	1.134	+	0.866	
REPLY							1.002		**1.029**	∘	**1.059**	•
REPLYE							**4.808**	⋆	**5.073**	⋆	**7.708**	⋆
DLEN							**0.858**	⋆	**0.904**	⋆	0.948	
DLENE							**0.305**	⋆	0.541	+	**0.150**	⋆
INT							**1.210**	⋆	**1.186**	⋆	1.081	
INTE							**2.580**	⋆	**3.415**	⋆	**3.148**	⋆
WA									**0.728**	⋆	**0.744**	⋆
WAE									**0.342**	⋆	**0.257**	⋆
Chi Sq.	**1198.83**	⋆	**1355.13**	⋆	**1364.31**	∘	**1470.4**	⋆	**1521.04**	⋆	**3106.11**	⋆
Dev. Expl.	7.84%		8.86%		8.92%		9.62%		9.95%		20.31%	
Delta Chisq			156.3		9.18		106.09		50.64		1585.07	

For every stepwise model, we report odds ratios for each regression coefficient, as well as a goodness of fit metric (χ^2), the percentage of variation in the data each model explains (Dev. Expl) and the difference in goodness of fit compared to the previous hierarchical modelling step ($\Delta\chi^2$)

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

between the number of source code examples (NSOURCE) and post-release defects might be explained through the need for concrete examples when issues are more complex, or hard to locate and fix. Further investigation of the relationship between variability in interestingness (INTE) and risk of post-release defects is needed before we can attempt an explanation, and is part of our future work.

5 Case Study Two: Mozilla Firefox

5.1 Data Collection

For our case study on the Mozilla FIREFOX project, we obtained a snapshot of the concurrent version control (CVS) system of the Mozilla platform, and a snapshot of the BUGZILLA issue tracking system. The CVS system contains the development history of the Mozilla platform up to (but not including) version 3.5 of the FIREFOX browser. For the development of FIREFOX version 3.5 and higher, the Mozilla team switched to the Mercurial distributed version control system.

Overall, we collected a total of 567,595 issues that have been submitted to the BUGZILLA system between April 1997 and August 2010. The collected version control history contains a total of 664,626 changes (accounting for 217,919 transactions)

that have been carried out between April 1998 and August 2010. For the Mozilla FIREFOX project, crash logs were recorded in the form of *Talkback* traces between release 2.0 and 3.0. This proprietary crash reporting system was replaced in release 3.0 with an open-source version. These new crash logs are no longer collected in the issue tracking system. As a result, we have no stack trace measure available for our case study on the Mozilla FIREFOX project.

In order to link issue reports to transactions in the version control system, we use the same approach described in our case study on the ECLIPSE project (Section 4). However, we modified the set of keywords that point at bug identifiers to include those patterns that are specific to the Mozilla project. Overall, we were able to recover 165,342 links between source code files and issue reports.

Similar to our case study on the ECLIPSE project (Section 4), we collected measurements for a period of six months before the releases of FIREFOX 1.5 (released November 29th, 2005), FIREFOX 2.0 (released October 24th, 2006), and FIREFOX 3.0 (released June 17th, 2008). For each release, we also collected the amount of post-release defects for a period of six months after release. In the following, we present a detailed case study on FIREFOX 3.0, which is followed by a discussion and comparison of earlier releases.

5.2 Preliminary Analysis of Social Interaction Measures

For our case study on FIREFOX 3.0, we follow the same statistical approach described earlier in our case study on the ECLIPSE project (Section 4.2) and begin with a general analysis of the data in the form of descriptive statistics, followed by an analysis of multi-collinearity. Table 7 presents a summary of our metrics in the form of descriptive statistics. We again observe a relatively high amount of skew

Table 7 Descriptive statistics of social interaction measures in the Mozilla FIREFOX project

	Mean	SD	Min	Max	Skew
POST	0.21	0.79	0.00	22.00	7.89
CHURN	2.74	4.58	1.00	83.00	6.67
NSOURCE	6.13	14.86	0.00	130.00	5.08
NSCOM	0.65	0.83	0.00	13.00	2.18
NPATCH	0.00	0.03	0.00	1.00	25.88
PATCHS	0.00	0.04	0.00	1.00	23.96
NLINK	2.04	3.38	0.00	31.00	5.11
NPART	5.90	3.71	1.00	41.00	2.32
NDEVS	5.39	3.09	1.00	33.00	1.91
NUSERS	0.51	1.09	0.00	19.00	4.42
SNACENT	0.26	0.08	0.00	0.50	−0.39
NMSG	19.21	15.74	1.00	137.00	1.91
REPLY	53.82	92.05	0.00	2246.00	9.17
REPLYE	0.21	0.11	0.00	0.64	0.11
DLEN	1212.47	1330.70	2.00	13325.00	2.45
DLENE	0.31	0.09	0.00	0.52	−0.85
INT	9.70	8.61	0.00	106.00	2.30
INTE	0.12	0.18	0.00	1.00	1.45
WA	24.31	16.90	0.00	86.00	1.11
WAE	0.09	0.14	0.00	1.00	1.32

in the data, such that we will use standard log transformations of all metrics in the remainder of our analysis.

The results of our analysis of pair-wise correlations are presented in Fig. 5. We see that our data exhibits a high amount of multi-collinearity that we have to deal with before creating our statistical models. Especially notable are the observed correlations between the measures of the number of participants in a discussion (NPART, NDEVS, NUSERS) and most other metrics, as well as correlations between interestingness (INTE) and workflow (WAE), and the size of patches (PATCHS) and the number of patches submitted (NPATCH).

In order to resolve these multi-collinearity issues, we again perform a step-wise VIF analysis, starting with a regression model that contains all our measurements as independent variables and removing the variable with the highest VIF at each step, before re-evaluating our model.

The results of this analysis are presented in Table 8. Model 1 denotes our starting model, containing all variables. We observe that the number of messages in the discussion (NMSG) has the highest VIF value of 20.44. We remove NMSG from our model and re-evaluate the variance inflation factors (Model 2), at which point we observe the highest VIF measure of 19.07 for the number of patches submitted (NPATCH). We remove NPATCH from our model and re-compute all variance inflation factors. In Model 3, we observe two variables with a VIF measure above 10: number of developers (NDEV) and the measure of centrality in the social network (SNACENT), which both showed a high inter-correlation in the correlogram (Fig. 5). We remove the variable with the higher VIF measure, NDEV at a value of 14.03, and re-compute variance inflation factors of the remaining variables

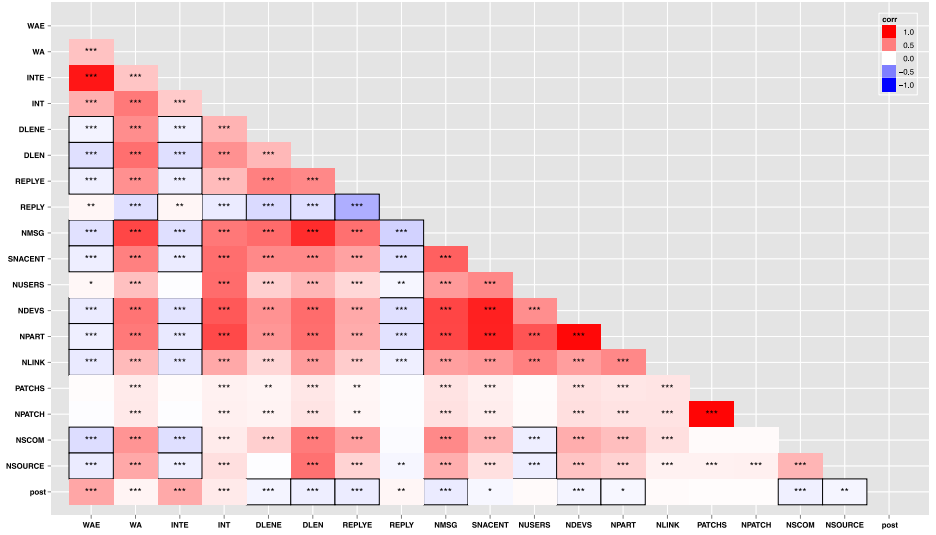

Fig. 5 Pairwise correlations of social interaction metrics in Mozilla FIREFOX 3.0 with levels * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. Strength of correlations is indicated by *colour* intensities; negative correlations are marked with an *outline*. The row labeled "post" refers to post-release defects

Table 8 Step-wise analysis of multicollinearity in the MOZILLA 3.0 dataset

$\log(Y_i)$	Variance inflation factor			
	Model 1	Model 2	Model 3	Model 4
NSOURCE	2.90	2.87	2.87	2.86
NSCOM	2.60	2.67	2.67	2.65
NPATCH	18.81	**19.07**	—	—
PATCHS	18.75	19.00	1.05	1.03
NLINK	1.88	1.89	1.89	1.86
NDEVS	15.45	14.05	**14.03**	—
NUSERS	2.29	2.22	2.22	1.99
SNACENT	14.01	13.45	13.45	2.92
NMSG	**20.44**	—	—	—
REPLY	1.43	1.36	1.36	1.33
REPLYE	2.20	2.33	2.33	2.30
DLEN	8.91	3.01	3.01	2.99
DLENE	4.14	1.84	1.84	1.83
INT	3.73	3.71	3.71	3.40
INTE	5.74	5.92	5.92	5.88
WA	3.85	2.95	2.95	2.92
WAE	6.88	7.12	7.12	7.12

Bold entries denote variables that have been selected for removal during VIF analysis (highest VIF measures in each model)

(Model 4). As all remaining variables have a VIF measure below 10, we stop at this point and obtain our final set of variables with minimized multicollinearity.

5.3 Hierarchical Analysis

Using the reduced set of regression variables that we obtained by resolving multicollinearity issues through step-wise VIF analysis in Section 5.2, we continue our hierarchical analysis to determine the relative impact of each of the four dimensions of social interaction metrics on post-release defects.

The results of our step-wise hierarchical analysis are presented in Table 9. Similar to our case study on ECLIPSE, we report all coefficients in the form of odds ratios, rather than the actual regression coefficients themselves, to ease interpretation.

Our hierarchical analysis starts with a baseline model MB, which relates only code churn to post-release defects. The results show, that churn is positively associated to post-release defects, same as it was in the ECLIPSE project.

Model M1 introduces the first dimension, structural information present in the discussions. Two information items turn out as statistically significant in the model: the amount of source code present in the discussion (NSOURCE), with an odds ratio of 0.891, and the amount of links (NLINK) with an odds ratio of 1.196. In contrast to our case study on ECLIPSE 3.0, NSOURCE is connected with an odds ratio smaller than 1.0, indicating that a larger amount of source code in the discussion is connected with a lower chance of post-release defects. A manual inspection of one hundred issue reports containing source code yielded no clear evidence on why this connection is opposite to our findings for ECLIPSE 3.0. On the other hand, the number of links in the discussion (NLINK), is connected with a larger amount of post-release defects, as it was in our case study on ECLIPSE 3.0. One possible explanation for this finding could be the presence of links to the external Talkback crash-report tracking systems present in the discussions of issue reports.

Table 9 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for Mozilla FIREFOX 3.0

log(Y_i)	MB		M1		M2		M3		M4		M5	
CHURN	**4.452**	⋆	**4.605**	⋆	**4.737**	⋆	**5.097**	⋆	**5.316**	⋆	**4.009**	⋆
NSOURCE			**0.891**	∘	0.904		0.943		1.018		0.915	
NSCOM			0.794		0.957		1.202		1.295		**1.599**	∘
PATCHS			4.974		8.498		6.168		**6.884**	∘	42.806	
NLINK			**1.196**	•	**1.378**	⋆	**1.508**	⋆	**1.475**	⋆	**1.253**	•
NPART					**0.177**	⋆	0.677		1.075		0.686	
NUSERS					**1.637**	⋆	**1.931**	⋆	**1.996**	⋆	**1.576**	•
SNACENT					**1208.985**	•	44.281		2.050		67.204	
REPLY							**0.928**	∘	0.987		1.000	
REPLYE							**0.016**	⋆	**0.067**	⋆	0.247	
DLEN							1.026		**1.117**	∘	1.043	
DLENE							0.814		**5.571**	•	1.150	
INT							**0.507**	⋆	**0.557**	⋆	0.846	
INTE							**3.061**	•	**3.969**	•	2.836	
WA									**0.492**	⋆	**0.715**	⋆
WAE									1.953		4.246	
χ^2	**744.14**	⋆	**773.88**	⋆	**807.54**	⋆	**933.29**	⋆	**1007.11**	⋆	**1836.43**	⋆
Dev. Expl.	14.90%		15.49%		16.17%		18.68%		20.16%		36.76%	
$\Delta\chi^2$			29.74		33.66		125.75		73.82		829.32	

For every stepwise model, we report odds ratios for each regression coefficient, as well as a goodness of fit metric (χ^2), the percentage of variation in the data each model explains (Dev. Expl) and the difference in goodness of fit compared to the previous hierarchical modelling step ($\Delta\chi^2$)

Bold entries denote significant regression coefficients (p level smaller than 0.05)

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

Summary: *Overall, our results indicate that for FIREFOX 3.0, source code in a discussion is connected with a decreased failure proneness, whereas the number of links is connected with an increase failure proneness. The explanatory power of the model increases by only 0.59% over the baseline model, however this increase is deemed statistically significant through analysis of variance (ANOVA).*

Model M2 introduces the second dimension of social interaction metrics: social structures. Similar to our findings in our previous case study on ECLIPSE 3.1 and ECLIPSE 3.2, we find a statistically significant connection of this dimension with post-release defects. All three variables, the number of participants in a discussion (NPART), the amount of participants that are considered users (NUSERS) and the centrality measure, which describes the degree to which participants communicate with everyone else in the discussion (SNACENT), are considered statistically significant for this model. Considering the odds ratios of each variable, we find that the overall number of participants in the discussion is connected with a decrease of post-release defects, whereas both the amount of users in the discussion as well as the degree to which each participant talks to everyone else, connected with an increase of post-release defects.

Summary: *Overall, we find the second dimension, social structures to have a statistically significant effect on the explanatory power of our model. Even though the*

relative increase in explanatory power over model M1 is only 0.68%*, this minor increase is statistically significant.*

Model M3 introduces the third dimension of metrics: communication dynamics. The results show a statistically significant and negative relation between the measures of reply time (REPLY) and the corresponding entropy measure (REPLYE), as well as interestingness of an issue report (INT). The variability in interestingness (INTE) shows the same statistically significant and positive relation to the risk of post-release defects, that we have observed earlier in our case study on ECLIPSE.

Summary: *Overall, we observe that the introduction of the third dimension (communication dynamics) leads to a four to five times increase of explanatory power in the model compared to the previous two dimensions (+2.51%). This increase is statistically significant. Similar to our findings in ECLIPSE, discussion flow and interestingness are strongly connected to post-release defects.*

Model M4 introduces the fourth dimension of social interaction metrics used in our study: workflow activity. In contrast to our findings on ECLIPSE 3.0, our results show a statistically significant, strongly negative relation between the amount of workflow activities (WA) and post-release defects in FIREFOX 3.0. Upon

Table 10 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for FIREFOX 1.5

log(Y_i)	MB		M1		M2		M3		M4		M5	
CHURN	**6.437**	⋆	**6.000**	⋆	**6.114**	⋆	**5.250**	⋆	**4.761**	⋆	**4.618**	⋆
NSOURCE			**1.162**	⋆	**1.104**	∘	**1.159**	•	**1.161**	•	**1.189**	•
NSCOM			**1.320**	⋆	**1.780**	⋆	**2.407**	⋆	**2.378**	⋆	1.279	
PATCHS			0.568		0.477		0.676		0.731		0.360	
NLINK			**1.073**	∘	**1.348**	⋆	**1.267**	⋆	**1.265**	⋆	1.046	
NPART					**0.474**	⋆	0.682		**0.641**	+	**1.938**	+
NUSERS					**1.150**	∘	1.055		1.068		**0.662**	⋆
SNACENT					2.931		7.403		**14.568**	+	0.140	
REPLY							**1.054**	∘	**1.050**	∘	1.048	
REPLYE							**0.397**	+	**0.384**	∘	**0.055**	⋆
DLEN							**0.856**	⋆	**0.879**	•	1.040	
DLENE							**0.075**	⋆	**0.081**	⋆	**0.046**	⋆
INT							**0.907**	∘	**0.888**	∘	**0.716**	⋆
INTE							**4.148**	⋆	**2.212**	∘	**4.004**	∘
WA									0.966		0.945	
WAE									**4.190**	•	**4.672**	∘
Chi-Sq	1269.46	⋆	1365.92	⋆	1432.34	⋆	1583.29	⋆	1593.34	⋆	2760.34	⋆
Dev.Expl	14.75%		15.87%		16.64%		18.40%		18.51%		32.07%	
Delta Chisq			96.46		66.42		150.95		10.05		1167	

For every stepwise model, we report odds ratios for each regression coefficient, as well as a goodness of fit metric (χ^2), the percentage of variation in the data each model explains (Dev. Expl) and the difference in goodness of fit compared to the previous hierarchical modelling step ($\Delta\chi^2$)

Bold entries denote statistically significant coefficients with a p level lower than 0.05

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

manual inspection of the workflow activities in FIREFOX, we found a large amount of supporting workflow activities, such as the addition of attachments, modification or addition of supporting meta-information, such as version information, keywords, quality assurance contacts, or testing and debugging information. In contrast, we found a relatively strong emphasis on actual process activities such as the re-assignment of the issue to a different developer, and other status changes that commonly relate to bug tossing (Guo et al. 2011) in ECLIPSE.

Summary: *Overall, the introduction of workflow activity metrics increases the explanatory power of the model by 1.48%. This increase is statistically significant. Our findings indicate a strong negative relation between workflow activities and post release defects.*

5.4 Additional Versions of Firefox

We repeated the same hierarchical modelling approach for two additional releases of FIREFOX. The final models are presented in Tables 10 and 11. Analysis of variance inflation factors led to the exclusion of NMSG and NDEVS in FIREFOX 1.5, and the exclusion of NMSG, NDEVS, as well as NPATCH in FIREFOX 2.0. As a

Table 11 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for FIREFOX 2.0

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**4.336**	⋆	**4.328**	⋆	**4.450**	⋆	**2.754**	⋆	**2.643**	⋆	**4.162**	⋆
NSOURCE			0.973		0.966		1.018		1.019		1.073	
NSCOM			1.121		1.107		**1.548**	•	**1.548**	•	1.041	
PATCHS			2.219		2.155		3.258		3.425		7.294	
NLINK			**1.084**	∘	**1.185**	⋆	**1.208**	⋆	**1.198**	⋆	0.967	
NPART					1.051		**0.592**	+	**0.610**	+	0.780	
NUSERS					**0.607**	⋆	**0.678**	⋆	**0.685**	⋆	**0.651**	∘
SNACENT					1.320		**74.664**	∘	**81.532**	∘	3.180	
REPLY							**1.036**	+	**1.034**	+	**1.089**	∘
REPLYE							**0.446**	+	**0.493**	+	0.399	
DLEN							**0.829**	⋆	**0.860**	⋆	**0.866**	∘
DLENE							**0.134**	⋆	**0.198**	⋆	0.848	
INT							**1.302**	⋆	**1.254**	⋆	**1.411**	•
INTE							**10.095**	⋆	**4.948**	⋆	**5.228**	•
WA									**0.912**	+	**0.871**	+
WAE									**4.380**	•	2.261	
ChiSq	**1292.6**	⋆	1298.99		**1347.93**	⋆	**1605.97**	⋆	**1621.35**	⋆	**3942.5**	⋆
Dev.Epl.	12.67%		12.74%		13.22%		15.75%		15.90%		39.15%	
DeltachiSq			29.74		33.66		125.75		73.82		829.32	

For every stepwise model, we report odds ratios for each regression coefficient, as well as a goodness of fit metric (χ^2), the percentage of variation in the data each model explains (Dev. Expl) and the difference in goodness of fit compared to the previous hierarchical modelling step ($\Delta\chi^2$)

Bold entries denote statistically significant coefficients with a p level lower than 0.05

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

result, NPATCH was only available for FIREFOX 1.5, where is was not deemed statistically significant, and NDEVS was not available for any release of FIREFOX.

Overall, we observe a similar increase in explanatory power for each dimension, as seen in the detailed case study on FIREFOX 3.0. This is especially true for the large increase in explanatory power, when including experience metrics in models M5. We find the following metrics consistent across all three releases of FIREFOX: CHURN, NSOURCE, NSCOM, PATCHS, NLINK, SNACENT, INTE and WAE are statistically significant and connected with an increased risk of post-release defects. REPLYE and WA are statistically significant and connected with a decreased risk of post-release defects. On the whole, we observe a larger set of consistent metrics for FIREFOX, than in the case of ECLIPSE.

6 Discussion of the Results of Both Case Studies

In this section, we present a summary and comparison of our findings from both case studies. For the purpose of comparing the regression variables of each model, we cannot directly compare the odds ratios, as discussed in Section 3. Instead we follow a standard approach in statistics, which has been successfully used in previous research (Mockus et al. 2009): to gain a comparable notion of the relative effect of each regression variable in each model, we first compute the mean values of each regression variable across the whole population from which the regression model was built. We use these mean values as an input into the linear equations connected with model $M4$ in each case study. This equation has the form $Y = \beta_0 + \beta_1 v_1 \ldots \beta_n v_n$, with β_i the regression coefficients reported earlier, and v_i the mean values for each regression variable. For each regression variable, we then increase that one variable by 20%, keeping all other regression variables constant, and obtain a value Y_E. The difference $\Delta_Y = Y_E - Y$ describes the relative effect of each regression variable on post-release defect probability within its respective range. Table 12 presents the results of this analysis for each project and release. To increase readability, we have ordered the regression variables for each model according to Δ_Y in decreasing order.

To further enable a simpler comparison of the effect of regression variables on post-release defect probability across releases and projects, we summarize our observations from Table 12 in Table 13. Regression variables that are statistically significant at $p < 0.005$ are marked in bold font face. When a regression variable increased the probability of post-release defects we put the value "POS", whenever a regression variable decreased the probability of post-release defects, we put the value "NEG". Variables that were removed by VIF analysis or were not available in a project (such as TRACES and NTRACE in FIREFOX), are left blank.

Overall, we note that only a few variables show a statistically significant connection to post-release defects across all releases and projects. In particular, these variables are: CHURN, NLINK, REPLYE, and INT. We also observe a number of regression variables, that are deemed statistically significant for one project but not for the other. In particular these variables are: NSCOM in FIREFOX, SNACENT in ECLIPSE, DLEN in ECLIPSE, INTE in FIREFOX, WA in ECLIPSE, and WAE in ECLIPSE.

Apart from statistical significance, the more interesting observation is the direction of effect, each variable has on the probability of post-release defects. We

observe, that CHURN, NSOURCE, PATCHS and INTE are consistently connected with an increased risk of post-release defects across all releases of both projects. For both, code churn (CHURN) and number of files modified by patches (PATCHS), these findings are in line with previous work (Hassan 2009; Nagappan and Ball 2007). However, the relationships between number of source code examples and post-release defects, as well as interestingness entropy and post-release defects are neither obvious, nor intuitive, and open research opportunities for future work. Overall, we would like to note, that these variables (in addition to code churn) might be valuable for the use in defect prediction across releases and projects.

Table 12 Relative effect of a 20% increase of a predictor variable on post-release defect probability

Delta Y	Variable	Delta Y	Variable
Eclipse 3.0		Firefox 1.5	
0.218179	NPART	0.222083	CHURN
0.179844	CHURN	0.050918	NSCOM
0.059674	WA	0.027278	WAE
0.050326	NSOURCE	0.026945	NLINK
0.028323	WAE	0.024222	NSOURCE
0.016858	NLINK	0.019508	INTE
0.008306	DLENE	0.009344	REPLY
0.00775	NSCOM	0.003484	SNACENT
0.006909	PATCHS	0.000899	PATCHS
0.000314	INTE	0.000716	NUSERS
−0.001661	TRACES	−0.001256	NPATCH
−0.002524	REPLY	−0.004555	WA
−0.003112	NTRACE	−0.023008	DLEN
−0.006269	NPATCH	−0.025847	REPLYE
−0.019646	DLEN	−0.026162	INT
−0.028327	NUSERS	−0.117742	DLENE
−0.028965	INT		
−0.045887	REPLYE		
−0.132794	NDEVS		
Eclipse 3.1		Firefox 2.0	
0.151672	CHURN	0.139716	CHURN
0.088716	WA	0.064652	SNACENT
0.078401	DLENE	0.040164	INTE
0.067766	NUSERS	0.031841	INT
0.026875	WAE	0.029121	WAE
0.020078	REPLYE	0.022206	NSCOM
0.018092	NSOURCE	0.021536	NLINK
0.010273	TRACES	0.005882	REPLY
0.009493	INT	0.002539	NSOURCE
0.002724	INTE	0.00018	PATCHS
0.000361	NPATCH	−0.016856	WA
0.000243	PATCHS	−0.018102	REPLYE
−0.00562	NTRACE	−0.026412	NUSERS
−0.015629	REPLY	−0.027568	DLEN
−0.017509	NSCOM	−0.071109	DLENE
−0.049688	SNACENT		
−0.06054	DLEN		
−0.101094	NLINK		

Table 12 (continued)

Delta Y	Variable	Delta Y	Variable
Eclipse 3.2		Firefox 3.0	
0.200102	CHURN	0.228522	CHURN
0.043769	NSOURCE	0.079335	DLENE
0.030726	REPLYE	0.048993	NLINK
0.023857	INT	0.045122	NUSERS
0.021552	INTE	0.029235	SNACENT
0.017061	TRACES	0.028377	INTE
0.005456	NUSERS	0.020166	DLEN
0.005128	REPLY	0.019694	NSCOM
0.002244	NLINK	0.011393	NPART
0.000235	PATCHS	0.011063	WAE
−0.004347	NTRACE	0.002835	NSOURCE
−0.015711	NSCOM	0.000679	PATCHS
−0.018358	DLEN	−0.002309	REPLY
−0.022822	WAE	−0.091363	REPLYE
−0.023887	DLENE	−0.097416	INT
−0.032865	NDEVS	−0.124795	WA
−0.051129	WA		

Furthermore, we observe a number of regression variables that are consistent within one project. In particular, these variables are: NSCOM, NLINK, SNACENT, REPLYE, WA and WAE for FIREFOX, as well as NDEVS, DLEN, and SCNACENT for ECLIPSE. These variables might still be valuable for the use in defect prediction within the same project, across multiple releases of the software.

Table 13 Summary of effect direction of each regression variable on the probability of post-release defects

	Mozilla			Eclipse		
	v1.5	v2.0	v3.0	v3.0	v3.1	v3.2
CHURN	**POS**	**POS**	**POS**	**POS**	**POS**	**POS**
NSOURCE	**POS**	POS	POS	**POS**	**POS**	**POS**
NSCOM	**POS**	**POS**	**POS**	POS	**NEG**	**NEG**
NPATCH	NEG			**NEG**	POS	
PATCHS	POS	POS	**POS**	**POS**	POS	POS
NTRACE				NEG	NEG	NEG
TRACES				NEG	**POS**	**POS**
NLINK	**POS**	**POS**	**POS**	**POS**	**NEG**	**POS**
NUSERS	POS	**NEG**	**POS**	NEG	**POS**	**POS**
NDEVS				NEG		**NEG**
SNACENT	**POS**	**POS**	POS		**NEG**	
DLEN	**NEG**	**NEG**	POS	**NEG**	**NEG**	**NEG**
DLENE	**NEG**	**NEG**	POS	POS	**POS**	**NEG**
REPLY	**POS**	**POS**	**NEG**	NEG	**NEG**	**POS**
REPLYE	**NEG**	**NEG**	**NEG**	**NEG**	**POS**	**POS**
INT	**NEG**	**POS**	**NEG**	**NEG**	**POS**	**POS**
INTE	**POS**	**POS**	**POS**	POS	**POS**	**POS**
WA	NEG	**NEG**	**NEG**	**POS**	**POS**	**NEG**
WAE	**POS**	**POS**	POS	**POS**	**POS**	**NEG**

Statistically significant regression variables at $p < 0.005$ are marked in bold font

In addition to the particular relationships between single regression variables and the probability of post-release defects, we observe a number of consistent trends for our overall prediction models. In particular, the four dimensions of social interaction metrics are able to improve the explanatory power of models between 2.17 (FIREFOX 1.5) and 3.08 (FIREFOX 2.0) times the power of the baseline models, built on code churn. Second, each dimension adds a statistically significant amount of explanatory power of the previous dimensions (except for dimension two, social structures in the case studies of FIREFOX 2.0 and ECLIPSE 3.0).

6.1 Discussion of Entropy Measures

In order to interpret the observations for variables, which capture the variability of particular metrics across different observations (messages, discussions, files), we discuss each such variable in more detail in the remainder of this section.

For each assessment of entropy metrics, we first split the original dataset from which we built the regression models used in our case studies into two parts: one part contains all the files that had at least one post-release defect in the period of 6 months after release; the other part contains those that had none. For each part, we then collect the measurements of entropy and compare both distributions of measurements using an unpaired two-sided non-parametric statistical test, called the Mann-Whitney-U test, which is more robust against outliers than classical test, such as Student's t-test and does not rely on the assumption that the underlying data has normal distribution (Fay and Proschan 2010). The results of this analysis are summarized in Table 14 and discussed in the following.

6.1.1 Discussion of Entropy in Discussion Length (DLENE)

For both projects, we consistently observe a higher mean entropy for discussion length in files that had no post-release defects, than in files that had post-release defects. Except for ECLIPSE 3.0, the Mann-Whitney test confirms that this difference is statistically significant at $p < 0.005$.

Summary: *For ECLIPSE and FIREFOX, files that have no post-release defects exhibit more consistency in the length of the discussions on issue reports, connected to those files. Hence, we conclude that a larger variability in wordiness of discussions is connected to an increased risk of post-release defects.*

Table 14 Summary of entropy measure analysis for ECLIPSE and FIREFOX

Metric	Eclipse 3.0	Eclipse 3.1	Eclipse 3.2	Firefox 1.5	Firefox 2.0	Firefox 3.0
DLENE	—	$\mu_1 > \mu_2$	$\mu_1 > \mu_2$	$\mu_1 > \mu_2$	$\mu_1 > \mu_2$	$\mu_1 > \mu_2$
REPLYE	$\mu_1 > \mu_2$	—	$\mu_1 > \mu_2$	—	$\mu_1 > \mu_2$	$\mu_1 > \mu_2$
INTE	—	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$
WAE	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$	$\mu_1 < \mu_2$

μ_1 denotes the mean entropy for files with no post-release defects; μ_2 denotes the mean entropy for files with post-release defects; — denotes Mann-Whitney test did not reject H_0 at $p < 0.005$

6.1.2 Discussion of Entropy in Reply Time (REPLYE)

For both projects, we consistently observe a statistically significant connection between the variability in reply times between discussion messages, and post-release defects. For FIREFOX 2.0 and FIREFOX 3.0, as well as for ECLIPSE 3.0 and ECLIPSE 3.2, we find that the mean value of reply time entropy is higher for files that had no post-release defects, than for files that had post-release defects. Except for FIREFOX 1.5 and ECLIPSE 3.1, the Mann-Whitney test, confirms that the differences in both projects are statistically different at $p < 0.005$.

Summary: *Since a lower measure of entropy value is connected to a greater spread of values, we conjecture that for both projects outliers in reply time, i.e., a significant delay in the flow of the discussion, is connected to a larger risk of post-release defects. Our findings thus indicate that inconsistencies in information flow stand in relation to post-release defects.*

6.1.3 Discussion of Entropy in Interestingness (INTE)

For both projects, we observe that the mean entropy for interestingness is lower in files that had no post-release defects, than in files that had post-release defects. Except for ECLIPSE 3.0, the Mann-Whitney test confirms that the difference is statistically significant at $p < 0.005$.

Summary: *Files that have no post-release defects exhibit a larger variability of interestingness across all the issue reports connected to that file. At the same time, files that have post-release defects show a more consistent interestingness across all issue reports connected to that file. Hence, we conclude that a larger variability in interestingness is connected to a decreased risk of post-release defects.*

6.1.4 Discussion of Entropy in Workflow Activity (WAE)

We observed a statistically significant connection between workflow activity entropy and post-release defects for both, FIREFOX and ECLIPSE. Similarly, we find for both projects, a lower mean entropy for workflow activity in files that had no post-release defects, than in files that had post-release defects. A Mann-Whitney test confirms that the difference between both distributions is statistically significant at $p < 0.005$ in all cases.

Summary: *Files that have no post-release defects exhibit a larger variability of workflow activity across all the issue reports connected to that file. At the same time, files that have post-release defects show a more consistent workflow activity across all issue reports connected to that file. We hence conjecture that workflows involving a greater variety of steps are connected to an increased risk of post-release defects.*

7 Enhancing Traditional Models with Social Information

Through the two case studies presented in Sections 4 and 5, we have demonstrated that statistical models based on social interaction metrics yield an increase in

explanatory power of up to 16.36% (ECLIPSE) to 21.86% (FIREFOX) over the baseline model using code churn.

In this part of our analysis we want to investigate whether social information metrics can augment existing, top-performing defect prediction models that are based on an extensive set of source-code and file metrics. To perform this comparison, we use a publicly available defect prediction dataset, which was prepared by the University of Saarland (Zimmermann et al. 2007). As Bird et al. note (Bird et al. 2009), this dataset is extensively documented and has been widely used in research.

Among other information, this data set contains a variety of source-code and file-level metrics for files of different Eclipse releases. We are specifically interested in the latest release contained in this dataset, Eclipse 3.0, as we measured the social interaction metrics presented in this study during the same time period. Both datasets contain a different set of metrics for the same source code files: Zimmermann et al.'s dataset contains source code metrics, whereas our dataset contains social interaction metrics. We will use a combination of both datasets for the remainder of this section, to study the effects of combining both sources of information.

We first re-create the original code-metrics based model created by Zimmermann et al. (2007). The original statistical model M is presented in Table 15. This model contains the following regression variables: pre denotes the amount of defects associated with the file in the past 6 months before release. MLOC_max measures the maximum amount of non-blank, non-comment lines of source code inside method bodies. PAR measures the number of parameters inside method signatures, both as a maximum across all methods in a file, and as a sum of all methods. TLOC denotes the total lines of code in a file, including blank lines and comments.

We assess it using the same criteria as the models we derived from our hierarchical analysis. Our results show that the model has an explanatory power of $\chi^2 = 889.48$ (17.04% of deviance explained) and all regression variables are statistically significant at $p < 0.1$.

Next, we create an extended model M' by adding the set of social interaction metrics that we found statistically significant in our hierarchical analysis of ECLIPSE 3.0, to model M. This extended model is presented in Table 16. We observe that the addition of social interaction metrics increases the explanatory power of the new model M' by $\chi^2 = 716.55$ to a total of $\chi^2 = 1606.03$. This corresponds to an increase of 13.73% percent of additional deviance explained, to a total of 30.77%. An ANOVA test confirms that the observed increase over the baseline model M is statistically significant at $p < 0.001$. To compare, the best performing model based on source code metrics was reported by Shihab et al., with a total of 21.2% of deviance explained (Shihab et al. 2010b). By adding social interaction metrics, we are

Table 15 Baseline model M

	Estimate	Std. error	z value	Pr(>|z|)
(Intercept)	−4.7228	0.2475	−19.08	0.0000
log(1 + pre)	1.1766	0.0890	13.22	0.0000
log(1 + MLOC_max)	−0.2049	0.0606	−3.38	0.0007
log(1 + PAR_max)	0.4081	0.1324	3.08	0.0021
log(1 + PAR_sum)	−0.1599	0.0888	−1.80	0.0716
log(1 + TLOC)	0.8058	0.0936	8.60	0.0000

Table 16 Augmented model M'

	Estimate	Std. error	z value	Pr(>|z|)
(Intercept)	−5.2783	0.2860	−18.46	0.0000
log(1 + pre)	0.9341	0.0987	9.46	0.0000
log(1 + NSOURCE)	0.5128	0.0657	7.81	0.0000
log(1 + NPATCH)	−1.5056	0.7009	−2.15	0.0317
log(1 + PATCHS)	2.0683	0.9862	2.10	0.0360
log(1 + NLINK)	0.5146	0.1175	4.38	0.0000
log(1 + REPLYE)	−2.0122	0.5515	−3.65	0.0003
log(1 + WA)	0.3919	0.0796	4.92	0.0000
log(1 + MLOC_max)	−0.1839	0.0614	−3.00	0.0027
log(1 + PAR_max)	0.3795	0.1354	2.80	0.0051
log(1 + PAR_sum)	−0.1743	0.0908	−1.92	0.0550
log(1 + TLOC)	0.7939	0.0950	8.36	0.0000

able to greatly outperform this model, demonstrating the additional value of social information for defect prediction models.

Summary: *The large increase in explanatory power of the augmented model demonstrates that social interaction metrics are valuable for complementing traditional prediction models based on source-code based product and process metrics.*

8 Related Work

In the following, we discuss related research from the two major research areas of defect prediction and social analyses of software development. Several researchers have previously investigated the use of data captured from version control systems and bug databases for defect prediction. Basili et al. (1996) established and promoted the usefulness of object-oriented code metrics for predicting the defect density of code. Ohlsson and Ahlberg were among the first researchers to use code-oriented metrics to predict failure prone modules of a software (Ohlsson and Alberg 1996).

Extensive work by Nagappan and Ball (2005, 2007) has investigated the value of code and churn metrics to predict defects in large-scale commercial systems. Schroeter et al. showed that module dependencies, which are already available at design time can be used to predict software defects (Schröter et al. 2006).

Hassan demonstrates in a large case study that prediction models based on change complexity outperform traditional churn based prediction models (Hassan 2009). Zimmermann et al. use social network measures on dependency graphs to predict defects (Zimmermann and Nagappan 2008).

In contrast to previous research, the work presented in this study does not focus on formulating accurate prediction models. We rather focus on using statistical models and the insights about relationships between variables that can be gained from studying these models, to investigate the relationships between social interactions and software quality. As such, we use prediction models as an explorative tool in the same vein of work done by Mockus et al. (2005, 2009).

Shihab et al. (2010b) carried out an analysis of the variation inflation factors of the source-code based process and product metrics, initially reported for the ECLIPSE

project by Zimmermann et al. (2007). Our work extends this study by adding social interaction metrics to the defect prediction model presented in the study by Shihab et al.

The work by Wolf et al. (2009) presents a case study on the use of social network analysis measures obtained from inter-developer communication in the IBM Jazz repository to predict build failures. Similar use of socio-technical network measures to predict software defects has been carried out by Pinzger et al. (2008) and Meneely et al. (2008).

A related study was conducted by Bacchelli et al. (2010) that investigates the possible use of code popularity metrics obtained from email communication among developers for defect prediction. Our work differs from these studies, as we use a variety of measures of social information to study relationships between these measures and the strength of their associations rather than performing actual predictions.

A recent study by Jeong et al. uses workflow activity recorded in the issue tracking system of the ECLIPSE project to improve issue assignment. Together with the work of Guo et al. (2010, 2011) as well as Shihab et al. (2010a), empirical evidence on the possibly negative effects of workflow activity served as an intuition for including workflow activity metrics for defect modelling in our work.

9 Threats to Validity

We discuss the limitations of our study and the applicability of the results derived through our approach. For this purpose we discuss our work along four types of validity (Yin 1994): construct validity, internal validity, external validity and reliability.

9.1 Construct Validity

The assessment of construct validity aims at evaluating the meaningfulness of measurements and whether they quantify what we want them to. The conjecture of our work is that the social interactions between developers and users, which surround the development process of a software system has an impact on the quality of the final software product. In order to capture social interaction, we use issue tracking systems as a repository containing records of such interaction. We focus on issue tracking systems over less formal and structured repositories, such as mailing lists, as issue tracking system are well understood and contain a wealth of historic data surrounding the evolution and maintenance of a software system. Furthermore, they contain meta-data that allows us to reliably establish traceability links back to the source of a software system.

One big assumption of our work is that the source code and issue repositories capture all the data, which might generally not be the case. However, the quality and extend of data recorded in open-source projects is likely very high, as development teams in open-source are often distributed across different countries and timezones, and thus heavily depend on the completeness of data, for the success of their collaborative software development efforts.

At the core of interaction in a software community are discussions, such as the discussion on the reported issues recorded in issue tracking systems like BUGZILLA. With respect to issue report discussions, we defined a set of metrics along four

different conceptual dimensions. Our metrics of the first dimension, discussion contents are based on a previous survey of developers (Bettenburg et al. 2008a) and focus on quantifying the presence of those information items that developers regarded as most helpful when working with issue reports (i.e., fixing a software defect). We capture these information items through an automated approach, which has been evaluated in previous research (Bettenburg et al. 2008a, b).

Our metrics of the second dimension, social structures, assume that the actors in the issue tracking system can take on one of two roles: developer or user. We follow previous work in the area (Jeong et al. 2009; Śliwerski et al. 2005) and define a developer as an actor in the system who has been assigned to working on at least one issue in the past. However, there might be actors present in the system, who are developers, but have never been assigned to fixing a bug. Furthermore, we capture the expertise of participants through determining the past amount of contributed messages to discussions, as past research has demonstrated that issues reported by experts have a higher likelihood of being successfully closed (Guo et al. 2010). Our approach is limited by recording only the top three experts of each discussion. We chose a threshold of three, to keep the artificial inflation of our models through the introduction of 'dummy' variables low. In order to capture the structural properties of a discussion, we use a metric from social network analysis, closeness-centrality (Wasserman and Faust 1994), which captures the extent, to which a participant talks to every other participants. As closeness-centrality is a per participant metric, we aggregate over all participants by the use of normalized entropy. A healthy discussion, in which every participant interacts with every other participant would thus be characterized by maximum entropy. Our approach could explore the use of other metrics from the area of social network analysis, or aggregate per participant metrics differently.

Our metrics of the third dimension, communication dynamics, focus on the quantitative measurements of a discussion, such as the number of exchanged messages, their length, time and interestingness. Our approach approximates interestingness as the degree of exposure to actors in the bug tracking system. For this purpose we use the notification list, on which participants can sign up to be notified when the information on an issue changes. By design of the issue tracking system, every participant in a discussion is automatically put on the notification list. Furthermore, issues might be interesting to actors, but actors might decline to sign up on the notification list, e.g., to avoid additional email traffic.

Our metrics of the fourth dimension, workflow, strongly focusses on the workflow activities that are recorded by the issue tracking system (Čubranić and Murphy 2003), which follow the life of an issue report from creation to closure. However, we can not observe any workflow activity beyond those recorded in the issue tracking system. While some issue tracking systems, such as `FIREFOX` record an extensive set of workflow activities, our manual inspection of the data revealed that the workflow in the `ECLIPSE` bug tracking system is mostly concerned with activities related to issue management.

For all dimensions, our approach is limited by capturing only a small subset of possible metrics. Using the same line of work, we could extend the set of metrics of social interaction to accommodate and explore further hypotheses of relations between social interaction and post-release defects, beyond those presented in this work.

In order to judge whether the models M1 to M5 obtained through our hierarchical modelling approach describe valid relationships, rather than random observations

in the data, we have carried out tests to judge the possible overfitting of every statistical model (the test is described in detail in Section 4.3). The only instance of overfitting we have observed was caused by the inclusion of experience metrics CON1-CON3 in model M5 for both projects, and across all releases. As a result, we have removed the experience metric from our hierarchical modelling approach and performed comparisons of models across releases and projects, as well as the discussions of results based on the most complete model, which does not suffer from overfitting: Model M4.

9.2 Internal Validity

The assessment of internal validity deals with the concern that there may be other plausible hypotheses that explain our findings. Furthermore, can we show that there is a cause and effect relation between the processes captured through our metrics of social interaction, and post-release defects?

Our approach uses regression models to put a set of regression variables (our metrics of social interaction) into a relation with a dependent variable (post-release defects). The effect of each variable is described through odds ratios. However, odds ratios describe only the magnitude and direction a unit change of the independent variable will have on the dependent variable, while keeping all other variables constant. Second, we can only observe correlation through statistical models, not causation. As such, all observations that we report on, even though they describe statistically significant connections, denote that we observed a co-occurrence of certain properties. In order to investigate possible causal effects, we would need to carry out a root-cause analysis along each variable.

For each observation we attempt to give an intuitive rationale for that particular connection. Where applicable, we carried out a manual inspection of our collected data with a specific observation in mind. However, there may be plausible rival hypotheses to explain our observations. Our work builds the basis for future investigations, by identifying a number of statistically significant connections between social interactions and software quality. At the same time, some of our observations confirm findings of previous studies: code churn being closely tied to post-release defects (Nagappan and Ball 2007) and the spread of changes being connected to post-release defects (Hassan 2009).

9.3 External Validity

The assessment of external validity evaluates to which extent generalization from the results of our study are possible. We have performed two case studies on large-scale open-source software systems with different domains: `ECLIPSE` is an integrated development environment, `FIREFOX` is a web-browser. In addition, we have studied multiple releases of each software system to further reduce threats to external validity. However, since processes and practices differ greatly between open-source development and commercial development, our observations might not generalize to industrial settings. We believe, that the approach described in this paper, together with the provided resources, can be readily adopted to other projects, as long as these projects provide records that allow to establish traceability links between the source code and issue tracking system.

9.4 Reliability

The assessment of reliability of our study refers to the degree to which someone analyzing the data presented in this work would reach the same results or conclusions. We believe that the reliability of our study is very high. Our data is derived from publicly available source control and issue tracking systems. Additionally, Appendix A provides the underlying source code and datasets of all analyses carried out in this paper, to enable others to replicate and extend our work.

10 Conclusions and Future Possibilities

In this study we investigated the impact of social interaction metrics (measured from discussions on issues mined from issue tracking systems) on software quality, expressed through their impact on post-release defects. Our results illustrate the relationships between the regression variables contained in four different dimensions of social interactions metrics (discussion contents, social structures, communication dynamics, and workflow) and the risk of post-release defects. We expressed these relationships both, as odds ratios in logarithmic space, as well as their relative impact through a separate statistical analysis, to give the reader a better feeling for the direction and magnitude of the effect of each variable.

Our results not only confirm the consistent relationship between code churn and increased risk of post-release defects presented by previous research in this area, but also establish a set of metrics that might be equally valuable for defect prediction. In particular we found that the number of source code snippets discussed by developers and users (NSOURCE) are related to an increased risk of post-release defects. This finding is consistent across all releases and projects studied. Further research is needed to investigate the intricacies of this connection, e.g., are code examples needed for meaningful discussions of more complex or error-prone code changes Other metrics that were consistently connected to an increased risk of post-release defects include the spread of changes proposed through patches (PATCHS), the variability of interestingness (INTE), as well as variability of workflow (WAE).

At the same time, our study presented metrics that were consistent across several releases of the same project, but not across different projects. These include the overall number of stack traces discussed (NTRACES), information flow effectivity (SNACENT), and overall workflow activities (WA).

In addition, our findings demonstrate that a combination of both, source-metrics based models and social metrics based models yields a higher explanatory power than either of the models on its own. As a result we conjecture that models based on social interaction metrics not only explain a similar amount of deviance as traditional models, which use source-code based product and process metrics, but can be used to complement traditional models to obtain higher explanatory power than each model taken on its own.

We feel that our findings confirm the value of information about social interaction among community members of a software for software engineering research community. This establishes social information metrics as a promising direction to explore for future research in defect prediction and software maintenance.

Acknowledgements We want to thank Audris Mockus of Avaya Labs and the anonymous reviewers of ICPC '10 for their many helpful comments on earlier revisions of this study.

Appendix A: Repeatability of our Work

We have carried out most of our statistical analyses in a statistical tool called R, which is a widely used open-source implementation of the commercial S programming language, originally developed by John Chambers at Bell Labs. For further information and downloads, we direct our interested reader to http://cran.r-project.org/. In order to provide a common ground for future research in this area, and to enable repeatability of our work, we present the source code for all our analyses carried out in this study in this Appendix. In addition to the R scripts, we provide the datasets for both case studies under the following URL: http://sailhome.cs.queensu.ca/replication/social-interactions/.

Case Study One: ECLIPSE

```
#-------------------------------------------------------------
#    File:          Analysis_ECLIPSE.R
#    Date:          December, 2010

#-------------------------------------------------------------
# Load Required Libraries
library(Design)
library(BiodiversityR)
library(ggplot2)


#-------------------------------------------------------------
# Load ECLIPSE dataset
d1 <- read.csv('emse-data-eclipse-3.0.csv')

#-------------------------------------------------------------
# Descriptive Statistics (Table 2)
means <- lapply(d1[,-c(1,2,28,29,30)], FUN=mean)
sds <- lapply(d1[,-c(1,2,28,29,30)], FUN=sd)
maxs <- lapply(d1[,-c(1,2,28,29,30)], FUN=max)
mins <- lapply(d1[,-c(1,2,28,29,30)], FUN=min)
skews <- lapply(d1[,-c(1,2,28,29,30)], FUN=skewness)

# You can also investigate kurtosis by calling
# kurtoses <- lapply(d1[,-c(1,2,28,29,30)], FUN=kurtosis)

descriptive <- cbind(Mean=means, SD=sds, Min=mins,
                     Max=maxs, Skew=skews)

xtable(descriptive)
#-------------------------------------------------------------
# Correlogram (Figure 3)

# First, we create a helper function to record the p-values of
# the correlation tests
cor.pvalues <- function(X){
    nc <- ncol(X)
    res <- matrix(0, nc, nc)
```

```
    for (i in 2:nc){
        for (j in 1:(i - 1)){
            res[i, j] ← res[j, i] ← cor.test(X[,i], X[,j])
                $p.value
            }
        }
    res
}

# We save the actual correlation values in c
c ← cor(d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,20,22,23,24,
        27,33,34,37,38)])

# We use our new function to save the correlation p-values
    in p
p ← cor.pvalues(d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,20,
        22,23,24,27,33,34,37,38)])

# A helper function to translate p-levels into stars
stars ← as.character(symnum(p, cutpoints=c(0,0.001,0.01,
    0.05,1),
                        symbols=c('***', '**', '*', '' ),
                        legend=F))

# Transform data molten.d1 ← cbind(melt(c), stars) names(molten.d1) ←
c("M1", "M2", "corr", "pvalue") x ←
ggplot(molten.d1, aes(M1, M2, fill=corr))

mi.ids ← subset(molten.d1, M1 == M2)
mi.lower ← subset(molten.d1[lower.tri(c),], M1 != M2)
mi.upper ← subset(molten.d1[upper.tri(c),], M1 != M2)

# Now plot just these values, adding labels (geom_text)
(p1 ← x +  geom_tile(data=mi.lower) +
     geom_text(data=mi.lower, aes(label=paste(pvalue)))
)
meas ← as.character(unique(molten.d1$M2))

# Write to a PDF file pdf(file="figure_1.pdf", height=10,
 width=18) (p2 ← p1 + scale_colour_identity() +
     scale_fill_gradientn(
        colours= c("red", "white", "blue"),
        limits=c(1,-1)) +
     scale_x_discrete(
        limits=meas[length(meas):1]) + #flip the x axis
     scale_y_discrete(
        limits=meas)
)

dev.off()

#-------------------------------------------------------------
#-------------------------------------------------------------

# Stepwise Analysis of Variance Inflation Factors (VIF)

# (Table 3)

# Step one: Start with full model model ← glm(post>0 ~
log(1+NSOURCE) + log(1+NSCOM)
```

```
    + log(1+NPATCH) + log(1+PATCHS) + log(1+NTRACE)
    + log(1+TRACES) + log(1+NLINK) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+SNACENT) + log(1+NMSG)
    + log(1+REPLY) + log(1+REPLYE) + log(1+DLEN)
    + log(1+DLENE) + log(1+INT) + log(1+INTE)
    + log(1+WA) + log(1+WAE)
    , data=d1, family=binomial())

summary(model)
vif(model)

# Remove NMSG
model <- glm(post>0 ~ log(1+NSOURCE) + log(1+NSCOM)
    + log(1+NPATCH) + log(1+PATCHS) + log(1+NTRACE)
    + log(1+TRACES) + log(1+NLINK) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+SNACENT) + log(1+REPLY)
    + log(1+REPLYE) + log(1+DLEN) + log(1+DLENE)
    + log(1+INT) + log(1+INTE) + log(1+WA)
    + log(1+WAE)
    , data=d1, family=binomial())

summary(model)
vif(model)

# Remove SNACENT
model <- glm(post>0 ~ log(1+NSOURCE) + log(1+NSCOM)
    + log(1+NPATCH) + log(1+PATCHS) + log(1+NTRACE)
    + log(1+TRACES) + log(1+NLINK) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+REPLY) + log(1+REPLYE)
    + log(1+DLEN) + log(1+DLENE) + log(1+INT)
    + log(1+INTE) + log(1+WA) + log(1+WAE)
    , data=d1, family=binomial())

summary(model)
vif(model)

#-------------------------------------------------------------
# Stepwise Hierarchical Modelling (Table 4)
#
# Helper function to compute odds ratios
lreg.or <- function(model)
  {
   lreg.coeffs <- coef(summary(model))
   lci <- exp(lreg.coeffs[ ,1] - 1.96 * lreg.coeffs[ ,2])
   or <- exp(lreg.coeffs[ ,1])
   uci <- exp(lreg.coeffs[ ,1] + 1.96 * lreg.coeffs[ ,2])
   lreg.or <- cbind(lci, or, uci)
   lreg.or
}

# Build Baseline Model
baseline <- glm ( (post>0) ~ log(1+CHURN), data=d1,
    family=binomial())
summary(baseline)
deviancepercentage(baseline,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
        20,22,23,24,27,33,34,37,38)])
lreg.or(baseline)

# Add Dimension 1
```

```
model1 ← glm ( (post>0) ~ log(1+CHURN) + log(1+NSOURCE)
    + log(1+NTRACE) + log(1+NPATCH) + log(1+NSCOM)
    + log(1+PATCHS) + log(1+TRACES) + log(1+NLINK)
    , data=d1, family=binomial())
summary(model1)
deviancepercentage(model1,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19
              ,20,22,23,24,27,33,34,37,38)])

# Compare Model 1 to Baseline
anova(baseline,model1, test="Chisq")
lreg.or(model1)

# Add Dimension 2
model2 ← glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NTRACE) + log(1+NPATCH)
    + log(1+NSCOM) + log(1+PATCHS) + log(1+TRACES)
    + log(1+NLINK) + log(1+NPART) + log(1+NDEVS)
    + log(1+NUSERS)
    , data=d1, family=binomial())
summary(model2)
deviancepercentage(model2,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
              20,22,23,24,27,33,34,37,38)])

# Compare Model 2 to Model 1
anova(model1,model2, test="Chisq")
lreg.or(model2)

# Add Dimension 3
model3 ← glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NTRACE) + log(1+NPATCH)
    + log(1+NSCOM) + log(1+PATCHS) + log(1+TRACES)
    + log(1+NLINK) + log(1+NPART) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+REPLY) + log(1+REPLYE)
    + log(1+DLEN) + log(1+DLENE) + log(1+INT)
    + log(1+INTE)
    , data=d1, family=binomial() )
summary(model3)
deviancepercentage(model3,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
        20,22,23,24,27,33,34,37,38)])

# Compare Model 3 to Model 2
anova(model2,model3, test="Chisq")
lreg.or(model3)

# Add Dimension 4
model4 ← glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NTRACE) + log(1+NPATCH)
    + log(1+NSCOM) + log(1+PATCHS) + log(1+TRACES)
    + log(1+NLINK) + log(1+NPART) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+REPLY) + log(1+REPLYE)
    + log(1+DLEN) + log(1+DLENE)
    + log(1+INT) + log(1+INTE)
    + log(1+WA) + log(1+WAE)
    , data=d1, family=binomial() )
summary(model4)
deviancepercentage(model4,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
```

```
        20,22,23,24,27,33,34,37,38)])

# Compare Model 4 to Model 3
anova(model3,model4, test="Chisq")
lreg.or(model4)

# Add Dimension CON1–CON3
model5 ← glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NTRACE) + log(1+NPATCH)
    + log(1+NSCOM) + log(1+PATCHS) + log(1+TRACES)
    + log(1+NLINK) + log(1+NPART) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+REPLY) + log(1+REPLYE)
    + log(1+DLEN) + log(1+DLENE) + log(1+INT)
    + log(1+INTE) + log(1+WA) + log(1+WAE)
    + CON1 + CON2 + CON3
    , data=d1, family=binomial() )
summary(model4)
deviancepercentage(model5,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
        20,22,23,24,27,33,34,37,38)])

# Compare Model 5 to Model 4
anova(model4,model5, test="Chisq")
lreg.or(model5)

# Plot Odds Ratios for developers by developer index
# Figure 4
ors ← lreg.or(model5[,2])
plot(ors[18−999], type="l"
    , xlab="Developer Index"
    , ylab="Relative Impact (Odds Ratios)")

#−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
# Combination of Traditional Models and New Models
# Section 7
mtrad ← glm ( (post>0) ~ log(1+pre)
    + log(1+MLOC_max) + log(1+PAR_max)
    + log(1+PAR_sum) + log(1+TLOC)
    , data=d1, family=binomial() )
summary(mtrad)
deviancepercentage(mtrad, data=d1)

# Table 10
xtable(mtrad)

# Add new Metrics
maug ← glm ( (post>0) ~ log(1+pre)
    + log(1+NSOURCE) + log(1+NPATCH)
    + log(1+PATCHS) + log(1+NLINK)
    + log(1+REPLYE) + log(1+DLEN) + log(1+DLENE)
    + log(1+INT) + log(1+INTE)
    + log(1+WA)
    + log(1+MLOC_max) + log(1+PAR_max)
    + log(1+PAR_sum) + log(1+TLOC)
    + CON1 + CON2 + CON3
    , data=d1, family=binomial() )
summary(maug)
deviancepercentage(maug, data=d1)

# Table 10
```

```
xtable(maug)

# Compare mtrad to maug
anova(mtrad,maug, test="Chisq")

#-----------------------------------------------------------
# ANALYSIS OF ENTROPY (Section 6)

# Analysis of REPLYE
# Extract postrelease defects and replytime entropy from
    dataframe
no_post <- subset(d1[,c(4,23)], post=0)
yes_post <- subset(d1[,c(4,23)], post>0)

mean(no_post$REPLYE)
mean(yes_post$REPLYE)

# Perform a Mann-Whitney U test to see,

# whether difference in means is statistically significant
wilcox.test(no_post$REPLYE, yes_post$REPLYE,paired=FALSE)

#-----------------------------------------------------------
# Analysis of WAE
# Extract postrelease defects and replytime entropy from dataframe
no_post <- subset(d1[,c(4,38)], post=0)
yes_post <- subset(d1[,c(4,38)], post>0)

mean(no_post$WAE)
mean(yes_post$WAE)

# Perform a Mann-Whitney U test to see,
# whether difference in means is statistically significant
wilcox.test(no_post$WAE, yes_post$WAE, paired=FALSE)
```

Case Study Two: Firefox

```
#-----------------------------------------------------------
#    File:        Analysis_FIREFOX.R
#    Date:        December, 2010

#-----------------------------------------------------------
# Load data
d1 <- read.csv('output-moz-3.0-6months.csv')

#-----------------------------------------------------------

# Load Required Libraries
library(Design)
library(BiodiversityR)
library(ggplot2)


#-----------------------------------------------------------
# STEP 1: Descriptive Statistics (Table 5)
means <- lapply(d1[,-c(1,2,28,29,30)], FUN=mean)
sds <- lapply(d1[,-c(1,2,28,29,30)], FUN=sd)
maxs <- lapply(d1[,-c(1,2,28,29,30)], FUN=max)
mins <- lapply(d1[,-c(1,2,28,29,30)], FUN=min)
skews <- lapply(d1[,-c(1,2,28,29,30)], FUN=skewness)
```

```
descriptive ← cbind(Mean=means, SD=sds, Min=mins,
        Max=maxs, Skew=skews)

xtable(descriptive)
#-------------------------------------------------------------

#-------------------------------------------------------------
# Correlogram
# Figure 5
cor.pvalues ← function(X){
    nc ← ncol(X)
    res ← matrix(0, nc, nc)
    for (i in 2:nc){
        for (j in 1:(i − 1)){
            res[i, j] ← res[j, i] ← cor.test(X[,i],
                X[,j])$p.value
            }
        }
    res
}

# FIREFOX
c ← cor(d1[,c(4,6,7,9,10,15,16,17,18,19,
    20,22,23,24,27,33,34,37,38)])
p ← cor.pvalues(d1[,c(4,6,7,9,10,15,16,17,18,19,
    20,22,23,24,27,33,34,37,38)])

stars ← as.character(symnum(p, cutpoints=c(0,0.001,0.01,
    0.05,1),
                        symbols=c('***', '**', '*', '' ),
                        legend=F))

molten.d1 ← cbind(melt(c), stars) names(molten.d1) ←
c("M1", "M2", "corr", "pvalue") x ← ggplot(molten.d1,
aes(M1, M2, fill=corr))

mi.ids ← subset(molten.d1, M1 == M2)
mi.lower ← subset(molten.d1[lower.tri(c),], M1 != M2)
mi.upper ← subset(molten.d1[upper.tri(c),], M1 != M2)

# We now plot just these values, adding labels (geom_text)
(p1 ← x + geom_tile(data=mi.lower) +
     geom_text(data=mi.lower, aes(label=paste(pvalue)))
)
meas ← as.character(unique(molten.d1$M2))

pdf(file="correlations_firefox.pdf", height=10, width=18)

(p2 ← p1 + scale_colour_identity() +
     scale_fill_gradientn(
        colours= c("red", "white", "blue"),

        limits=c(1,−1)) +
     scale_x_discrete(
        limits=meas[length(meas):1]) + #flip the x axis
     scale_y_discrete(limits=meas)
)

dev.off()
```

```
#-------------------------------------------------------------
# VIF Analysis: FIREFOX
# Table 6
model <- glm(post>0 ~ log(1+NSOURCE) + log(1+NSCOM)
+ log(1+NPATCH) + log(1+PATCHS) + log(1+NLINK)
+ log(1+NDEVS) + log(1+NUSERS) + log(1+SNACENT)
+ log(1+NMSG) + log(1+REPLY) + log(1+REPLYE)
+ log(1+DLEN) + log(1+DLENE) + log(1+INT)
+ log(1+INTE) + log(1+WA) + log(1+WAE),
data=d1, family=binomial())

vif(model)

# REMOVE NMSG
model <- glm(post>0 ~ log(1+NSOURCE) + log(1+NSCOM)
+ log(1+NPATCH) + log(1+PATCHS) + log(1+NLINK)
+ log(1+NDEVS) + log(1+NUSERS) + log(1+SNACENT)
+ log(1+REPLY) + log(1+REPLYE) + log(1+DLEN)
+ log(1+DLENE) + log(1+INT) + log(1+INTE)
+ log(1+WA) + log(1+WAE), data=d1, family=binomial())
vif(model)

# REMOVE NPATCH
model <- glm(post>0 ~ log(1+NSOURCE) + log(1+NSCOM)
+ log(1+PATCHS) + log(1+NLINK) + log(1+NDEVS)
+ log(1+NUSERS) + log(1+SNACENT) + log(1+REPLY)
+ log(1+REPLYE) + log(1+DLEN) + log(1+DLENE)
+ log(1+INT) + log(1+INTE) + log(1+WA)
+ log(1+WAE), data=d1, family=binomial())
vif(model)

# REMOVE NDEVS
model <- glm(post>0 ~ log(1+NSOURCE) + log(1+NSCOM)
+ log(1+PATCHS) + log(1+NLINK) + log(1+NUSERS)
+ log(1+SNACENT) + log(1+REPLY) + log(1+REPLYE)
+ log(1+DLEN) + log(1+DLENE) + log(1+INT)
+ log(1+INTE) + log(1+WA) + log(1+WAE),
data=d1, family=binomial())
vif(model)

#-------------------------------------------------------------
# Stepwise Hierarchical Analysis begins below
# Table 7

# Helper function to compute odds ratios
lreg.or <- function(model)
  {
   lreg.coeffs <- coef(summary(model))
   lci <- exp(lreg.coeffs[ ,1] - 1.96 * lreg.coeffs[ ,2])
   or <- exp(lreg.coeffs[ ,1])
   uci <- exp(lreg.coeffs[ ,1] + 1.96 * lreg.coeffs[ ,2])
   lreg.or <- cbind(lci, or, uci)
   lreg.or
}

#-------------------------------------------------------------
# MOZILLA - Selected Measures
# names(d1[,c(4,6,7,10,15,16,18,19,22,23,24,27,33,34,
        37,38)])
```

```
#-----------------------------------------------------------------
# Build Baseline Model   MOZILLA
baseline <- glm ( (post>0) ~ log(1+CHURN), data=d1,
    family=binomial() )
summary(baseline)
deviancepercentage(baseline,
    data=d1[,c(4,6,7,10,15,16,18,19,
        22,23,24,27,33,34,37,38)])
lreg.or(baseline)

#-----------------------------------------------------------------
# Add Dimension 1
model1 <- glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NSCOM) + log(1+PATCHS)
    + log(1+NLINK)
        , data=d1, family=binomial() )
summary(model1)
deviancepercentage(model1,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19
        ,20,22,23,24,27,33,34,37,38)])

# Compare Model 1 to Baseline
anova(baseline,model1, test="Chisq")
lreg.or(model1)

#-----------------------------------------------------------------
# Add Dimension 2n
model2 <- glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NSCOM) + log(1+PATCHS)
    + log(1+NLINK)
    + log(1+NPART) + log(1+NUSERS) + log(1+SNACENT)
    , data=d1, family=binomial() )
summary(model2)
deviancepercentage(model2,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
        20,22,23,24,27,33,34,37,38)])

# Compare Model 2 to Model 1
anova(model1,model2, test="Chisq")
lreg.or(model2)

#-----------------------------------------------------------------
# Add Dimension 3
model3 <- glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NSCOM) + log(1+PATCHS)
    + log(1+NLINK)
    + log(1+NPART)  + log(1+NUSERS) + log(1+SNACENT)
    + log(1+REPLY) + log(1+REPLYE) + log(1+DLEN)
    + log(1+DLENE) + log(1+INT) + log(1+INTE)
    , data=d1, family=binomial() )
summary(model3)
deviancepercentage(model3,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19
        ,20,22,23,24,27,33,34,37,38)])

# Compare Model 3 to Model 2
anova(model2,model3, test="Chisq")
lreg.or(model3)
```

```
#-------------------------------------------------------------
# Add Dimension 4
model4 <- glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NSCOM) + log(1+PATCHS)
    + log(1+NLINK)
    + log(1+NPART) + log(1+NUSERS) + log(1+SNACENT)
    + log(1+REPLY) + log(1+REPLYE) + log(1+DLEN)
    + log(1+DLENE) + log(1+INT) + log(1+INTE)
    + log(1+WA) + log(1+WAE)
    , data=d1, family=binomial() )
summary(model4)
deviancepercentage(model4,
    data=d1[,c(4,6,7,9,10,12,13,15,16,17,18,19,
        20,22,23,24,27,33,34,37,38)])

# Compare Model 4 to Model 3
anova(model3,model4, test="Chisq")
lreg.or(model4)

#-------------------------------------------------------------
# Add CON1 - CON3
model5 <- glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NSCOM) + log(1+PATCHS)
    + log(1+NLINK)
    + log(1+NPART) + log(1+NUSERS) + log(1+SNACENT)
    + log(1+REPLY) + log(1+REPLYE) + log(1+DLEN)
    + log(1+DLENE) + log(1+INT) + log(1+INTE)
    + log(1+WA) + log(1+WAE) + CON1 +CON2 +CON3
    , data=d1, family=binomial() )

# Compare Model 5 to Model 4
anova(model4,model5, test="Chisq")
lreg.or(model5)

# Plot Odds Ratios for developers by developer index
# Figure 4
ors <- lreg.or(model5[,2])
plot(ors[18-999], type="l"
    , xlab="Developer Index"
    , ylab="Relative Impact (Odds Ratios)")

#-------------------------------------------------------------
# ANALYSIS OF ENTROPY (Section 6)

# REPLYE Analysis

# Extract postrelease defects and replytime entropy measures
from dataframe
no_post <- subset(d1[,c(4,23)], post=0)
yes_post <- subset(d1[,c(4,23)], post>0)

mean(no_post$REPLYE)
mean(yes_post$REPLYE)
# Perform a Mann-Whitney U test to see if the
# difference in means is statistically significant
boxplot(no_post$REPLYE, yes_post$REPLYE)
wilcox.test(no_post$REPLYE, yes_post$REPLYE, paired=FALSE)

#-------------------------------------------------------------
# INTE Analysis
```

```
# Extract postrelease defects and replytime entropy measures
from dataframe
no_post ← subset(d1[,c(4,34)], post=0)
yes_post ← subset(d1[,c(4,34)], post>0)

mean(no_post$INTE)
mean(yes_post$INTE)

# Perform a Mann–Whitney U test to see if the difference
# in means is statistically significant
boxplot(no_post$INTE, yes_post$INTE)
wilcox.test(no_post$INTE, yes_post$INTE, paired=FALSE)

#-------------------------------------------------------------
# DLENE Analysis

# Extract postrelease defects and replytime entropy measures
from dataframe
no_post ← subset(d1[,c(4,27)], post=0)
yes_post ← subset(d1[,c(4,27)], post>0)

mean(no_post$DLENE)
mean(yes_post$DLENE)

# Perform a Mann–Whitney U test to see if the difference
# in means is statistically significant
boxplot(no_post$DLENE, yes_post$DLENE)
wilcox.test(no_post$DLENE, yes_post$DLENE, paired=FALSE)
```

Modelling the Relative Effect of Regression Variables

```
mean_effect ← function(model, means) {
    Y ← predict(model, means)
    effects=list()
    for (i in 1:length(means)) {
        meansprime ← means
        meansprime[[i]] ← meansprime[[i]] * 1.20
        Yprime ← predict(model, meansprime)
        effects[names(meansprime[i])] ← (Yprime–Y)
    }
    return (effects)
}

# Eclipse 3.0
d1 ← read.csv('eclipse–6mo–30–avg_files.csv')
model_eclipse_30 ← glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NTRACE) + log(1+NPATCH)
    + log(1+NSCOM) + log(1+PATCHS) + log(1+TRACES)
    + log(1+NLINK) + log(1+NPART) + log(1+NDEVS)
    + log(1+NUSERS) + log(1+REPLY) + log(1+REPLYE)
    + log(1+DLEN) + log(1+DLENE)
    + log(1+INT) + log(1+INTE)
    + log(1+WA) + log(1+WAE)
    , data=d1, family=binomial() )

means ← lapply(d1[,–c(1,2,28,29,30)], FUN=mean)

t ← mean_effect(model_eclipse_30, means[3:38])
```

```
ut ← sort(unlist(t), decreasing=TRUE)
cat(sprintf("\t %f \t %s \n", ut, names(ut)))

# Eclipse 3.1
d1 ← read.csv('output_ECLIPSE_3_1.csv')
model_eclipse_31 ← glm(post>0 ~ log(1+CHURN)
                  + log(1+NSOURCE) + log(1+NSCOM)
                  + log(1+NPATCH) + log(1+PATCHS)
                  + log(1+NTRACE) + log(1+TRACES)
                  + log(1+NLINK) + log(1+NUSERS)
                  + log(1+SNACENT) + log(1+REPLY)
                  + log(1+REPLYE) + log(1+DLEN)
                  + log(1+DLENE) + log(1+INT)
                  + log(1+INTE) + log(1+WA)
                  + log(1+WAE), data=d1,
                      family=binomial())

means ← lapply(d1[,-c(1,2,28,29,30)], FUN=mean)

t ← mean_effect(model_eclipse_31, means)
ut ← sort(unlist(t), decreasing=TRUE)
cat(sprintf("\t %f \t %s \n", ut, names(ut)))

# Eclipse 3.2
d1 ← read.csv('output_ECLIPSE_3_2.csv')
model_eclipse_32 ← glm(post>0 ~ log(1+CHURN)
                  + log(1+NSOURCE) + log(1+NSCOM)
                  + log(1+PATCHS) + log(1+NTRACE)
                  + log(1+TRACES) + log(1+NLINK)
                  + log(1+NDEVS) + log(1+NUSERS)
                  + log(1+REPLY) + log(1+REPLYE)
                  + log(1+DLEN) + log(1+DLENE)
                  + log(1+INT) + log(1+INTE)
                  + log(1+WA) + log(1+WAE),
                      data=d1, family=binomial())

means ← lapply(d1[,-c(1,2,28,29,30)], FUN=mean)

t ← mean_effect(model_eclipse_32, means)
ut ← sort(unlist(t), decreasing=TRUE)
cat(sprintf("\t %f \t %s \n", ut, names(ut)))

# Firefox 1.5
d1 ← read.csv('output_MOZILLA_1_5.csv')
model_firefox_15 ← glm(post>0 ~ log(1+CHURN)
                  + log(1+NSOURCE) + log(1+NSCOM)
                  + log(1+NPATCH) + log(1+PATCHS)
                  + log(1+NLINK) + log(1+NUSERS)
                  + log(1+SNACENT) + log(1+REPLY)
                  + log(1+REPLYE) + log(1+DLEN)
                  + log(1+DLENE) + log(1+INT)
                  + log(1+INTE) + log(1+WA) + log(1+WAE),
                      data=d1, family=binomial())
means ← lapply(d1[,-c(1,2,28,29,30)], FUN=mean)

t ← mean_effect(model_firefox_15, means)
ut ← sort(unlist(t), decreasing=TRUE)
cat(sprintf("\t %f \t %s \n", ut, names(ut)))

# Firefox 2.0
```

```
d1 ← read.csv('output_MOZILLA_2_0.csv') model_firefox_20 ←
glm(post>0 ~ log(1+CHURN)
                + log(1+NSOURCE) + log(1+NSCOM)
                + log(1+PATCHS) + log(1+NLINK)
                + log(1+NUSERS) + log(1+SNACENT)
                + log(1+REPLY) + log(1+REPLYE)
                + log(1+DLEN) + log(1+DLENE)
                + log(1+INT) + log(1+INTE)
                + log(1+WA) + log(1+WAE),
                    data=d1, family=binomial())

means ← lapply(d1[,-c(1,2,28,29,30)], FUN=mean)

t ← mean_effect(model_firefox_20, means)
ut ← sort(unlist(t), decreasing=TRUE)
cat(sprintf("\t %f \t %s \n", ut, names(ut)))

# Firefox 3.0
d1 ← read.csv('output-moz-3.0-6months.csv') model_firefox_30 ←
glm ( (post>0) ~ log(1+CHURN)
    + log(1+NSOURCE) + log(1+NSCOM) + log(1+PATCHS)
    + log(1+NLINK) + log(1+NPART) + log(1+NUSERS)
    + log(1+SNACENT) + log(1+REPLY) + log(1+REPLYE)
    + log(1+DLEN) + log(1+DLENE) + log(1+INT)
    + log(1+INTE) + log(1+WA) + log(1+WAE),
        data=d1, family=binomial() )

means ← lapply(d1[,-c(1,2,28,29,30)], FUN=mean)

t ← mean_effect(model_firefox_30, means)
ut ← sort(unlist(t), decreasing=TRUE)
cat(sprintf("\t %f \t %s \n", ut, names(ut)))
```

Appendix B: Summary of Hierarchical Models with β-coefficients

Table 17 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for ECLIPSE 3.0

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**0.699**	⋆	**0.666**	⋆	**0.668**	⋆	**0.725**	⋆	**0.567**	⋆	**0.650**	⋆
NSOURCE			**0.229**	⋆	**0.230**	⋆	**0.248**	⋆	**0.248**	⋆	**0.222**	⋆
NTRACE			−0.102		−0.115		−0.063		−0.055		0.047	
NPATCH			**−0.680**	∘	**−0.678**	∘	**−0.547**	+	**−0.636**	∘	−0.536	
NSCOM			0.086		0.077		0.096		0.082		0.095	
PATCHS			**1.101**	∘	**1.101**	∘	**1.049**	∘	**1.105**	∘	**1.260**	•
TRACES			0.007		0.005		0.002		−0.005		−0.011	
NLINK			**0.246**	⋆	**0.208**	•	**0.204**	•	**0.222**	•	**0.203**	+
NPART					0.395		0.461		0.651		0.657	
NDEVS					−0.323		−0.235		−0.415		−0.562	
NUSERS					−0.126		−0.095		−0.160		−0.101	

Table 17 continued

log(Y_i)	MB		M1		M2	M3		M4		M5	
REPLY						0.008		−0.006		−0.008	
REPLYE						**−0.932**	⋆	**−1.086**	⋆	**−1.357**	⋆
DLEN						−0.029		**−0.047**	∘	**−0.057**	+
DLENE						0.398		0.097		0.310	
INT						**−0.081**	•	**−0.086**	•	−0.016	
INTE						0.045		0.006		0.116	
WA								**0.156**	⋆	**0.088**	+
WAE								**0.434**	∘	0.336	
CON1−3										Fig. 2	⋆
Chisq	**559.01**	⋆	**698.5**	⋆	700.15	**731.5**	⋆	**752.3**	⋆	**1055.19**	⋆
Dev.Expl.	10.71%		13.38%		13.41%	14.02%		14.41%		26.07%	
DeltaChisq			139.48		1.652	31.357		20.28		302.87	

Bold entries denote statistically significant coefficients with a p level lower than 0.1

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

Table 18 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for ECLIPSE 3.1

log(Y_i)	MB		M1		M2		M3		M4		M5	
CHURN	**1.883**	⋆	**1.744**	⋆	**1.735**	⋆	**1.659**	⋆	**1.192**	⋆	**1.354**	⋆
NSOURCE			**0.293**	⋆	**0.214**	•	**0.291**	⋆	**0.273**	⋆	**0.321**	⋆
NSCOM			**−0.489**	⋆	**−0.515**	⋆	**−0.491**	⋆	**−0.535**	⋆	**−0.429**	•
PATCHS			0.440		0.536		0.517		0.587		0.960	
NPATCH			0.230		0.049		0.470		0.372		0.597	
NTRACE			**−0.355**	+	**−0.442**	∘	**−0.404**	∘	−0.314		−0.251	
TRACES			**0.090**	+	**0.116**	∘	**0.146**	•	**0.087**	+	**0.095**	+
NLINK			**−1.296**	⋆	**−1.311**	⋆	**−1.377**	⋆	**−1.429**	⋆	**−1.044**	⋆
NUSERS					**1.247**	⋆	**1.444**	⋆	**1.617**	⋆	**1.358**	⋆
SNACENT					**−1.369**	⋆	**−1.943**	⋆	**−1.553**	•	**−1.404**	∘
REPLY							**−0.028**	+	**−0.086**	⋆	**−0.042**	∘
REPLYE							**1.284**	⋆	**0.942**	∘	**0.922**	∘
DLEN							**−0.274**	⋆	**−0.333**	⋆	**−0.284**	⋆
DLENE							**3.304**	⋆	**1.881**	⋆	**1.104**	∘
INT							**0.078**	+	0.068		**0.139**	∘
INTE							**0.435**	•	0.135		**0.320**	+
WA									**0.537**	⋆	**0.343**	⋆
WAE									**1.367**	⋆	**1.286**	⋆
CON1−3											n/a	⋆
χ^2	**1643.98**	⋆	**2134.46**	⋆	**2434.56**	⋆	**2644.3**	⋆	**2744.02**	⋆	**4288.3**	⋆
Dev. Expl.	11.66%		15.15%		17.27%		18.75%		19.46%		30.41%	
$\Delta\chi^2$			490.48		300.1		209.74		99.72		1544.28	

Bold entries denote statistically significant coefficients with a p level lower than 0.1

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

Table 19 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for ECLIPSE 3.2

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**1.529**	⋆	**1.453**	⋆	**1.461**	⋆	**1.204**	⋆	**1.582**	⋆	**1.759**	⋆
NSOURCE			**0.689**	⋆	**0.659**	⋆	**0.714**	⋆	**0.721**	⋆	**0.639**	⋆
NSCOM			**−0.558**	⋆	**−0.505**	⋆	**−0.573**	⋆	**−0.595**	⋆	**−0.720**	⋆
PATCHS			0.465		0.422		0.499		0.583		0.734	
NTRACE			**−0.544**	∘	**−0.568**	∘	−0.372		−0.389		−0.504	
TRACES			**0.201**	⋆	**0.203**	⋆	**0.154**	∘	**0.167**	•	**0.124**	+
NLINK			**0.226**	⋆	**0.201**	•	0.128		0.055		**0.433**	⋆
NDEVS					**−0.125**	∘	**−0.206**	•	**−0.243**	⋆	−0.141	
NUSERS					**0.170**	•	**0.119**	+	**0.125**	+	−0.144	
REPLY							0.002		**0.028**	∘	**0.058**	•
REPLYE							**1.570**	⋆	**1.624**	⋆	**2.042**	⋆
DLEN							**−0.153**	⋆	**−0.101**	⋆	−0.053	
DLENE							**−1.186**	⋆	**−0.614**	+	**−1.896**	⋆
INT							**0.191**	⋆	**0.170**	⋆	0.078	
INTE							**0.948**	⋆	**1.228**	⋆	**1.147**	⋆
WA									**−0.317**	⋆	**−0.296**	⋆
WAE									**−1.074**	⋆	**−1.358**	⋆
CON1−3											n/a	⋆
χ^2	**1198.83**	⋆	**1355.13**	⋆	**1364.31**	∘	**1470.4**	⋆	**1521.04**	⋆	**3106.11**	⋆
Dev. Expl.	7.84%		8.86%		8.92%		9.62%		9.95%		20.31%	
$\Delta\chi^2$			156.3		9.18		106.09		50.64		1585.07	

Bold entries denote statistically significant coefficients with a p level lower than 0.1

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

Table 20 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for Mozilla FIREFOX 1.5

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**1.862**	⋆	**1.791**	⋆	**1.810**	⋆	**1.658**	⋆	**1.560**	⋆	**1.530**	⋆
NSOURCE			**0.150**	⋆	**0.098**	∘	**0.147**	•	**0.150**	•	**0.173**	•
NSCOM			**0.277**	⋆	**0.576**	⋆	**0.878**	⋆	**0.866**	⋆	0.246	
PATCHS			−0.566		−0.740		−0.391		−0.314		−1.022	
NLINK			**0.070**	∘	**0.298**	⋆	**0.237**	⋆	**0.235**	⋆	0.045	
NPART					**−0.745**	⋆	−0.382		**−0.445**	+	**0.662**	+
NUSERS					**0.139**	∘	0.053		0.066		**−0.413**	⋆
SNACENT					1.07		2.002		**2.679**	+	−1.965	
REPLY							**0.053**	∘	**0.049**	∘	0.047	
REPLYE							**−0.923**	+	**−0.956**	∘	**−2.896**	⋆
DLEN							**−0.155**	⋆	**−0.129**	•	0.039	
DLENE							**−2.593**	⋆	**−2.519**	⋆	**−3.086**	⋆
INT							**−0.097**	∘	**−0.119**	∘	**−0.333**	⋆
INTE							**1.423**	⋆	**0.794**	∘	**1.387**	∘
WA									−0.035		−0.057	
WAE									**1.433**	•	**1.542**	∘
CON1−3												⋆
χ^2	**1269.46**	⋆	**1365.92**	⋆	**1432.34**	⋆	**1583.29**	⋆	**1593.34**	⋆	**2760.34**	⋆
Dev.Expl	14.75%		15.87%		16.64%		18.40%		18.51%		32.07%	
$\Delta\chi^2$			96.46		66.42		150.95		10.05		1167	

Bold entries denote statistically significant coefficients with a p level lower than 0.1

⋆ $p < 0.001$, • $p < 0.01$, ∘ $p < 0.05$, + $p < 0.1$

Table 21 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for Mozilla FIREFOX 2.0

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**1.467**	⋆	**1.465**	⋆	**1.493**	⋆	**1.013**	⋆	**0.972**	⋆	**1.426**	⋆
NSOURCE			−0.027		−0.035		0.018		0.019		0.070	
NSCOM			0.114		0.102		**0.437**	•	**0.437**	•	0.040	
PATCHS			0.797		0.768		1.181		1.231		1.987	
NLINK			**0.081**	○	**0.170**	⋆	**0.189**	⋆	**0.181**	⋆	−0.034	
NPART					0.050		**−0.524**	+	**−0.495**	+	−0.248	
NUSERS					**−0.499**	⋆	**−0.388**	⋆	**−0.379**	⋆	**−0.429**	○
SNACENT					0.278		**4.313**	○	**4.401**	○	1.157	
REPLY							**0.035**	+	**0.033**	+	**0.085**	○
REPLYE							**−0.808**	+	**−0.707**	+	−0.918	
DLEN							**−0.187**	⋆	**−0.151**	⋆	**−0.144**	○
DLENE							**−2.013**	⋆	**−1.618**	⋆	−0.165	
INT							**0.264**	⋆	**0.226**	⋆	**0.344**	•
INTE							**2.312**	⋆	**1.599**	⋆	**1.654**	•
WA									**−0.092**	+	**−0.138**	+
WAE									**1.477**	•	0.816	
χ^2	**1292.60**	⋆	1298.99		**1347.93**	⋆	**1605.97**	⋆	**1621.35**	⋆	**3942.50**	⋆
Dev. Expl.	12.67%		12.74%		13.22%		15.75%		15.90%		39.15%	
$\Delta\chi^2$			29.74		33.66		125.75		73.82		829.32	

Bold entries denote statistically significant coefficients with a p level lower than 0.1

⋆ $p < 0.001$, • $p < 0.01$, ○ $p < 0.05$, + $p < 0.1$

Table 22 Hierarchical analysis of logistic regression models along the four dimensions of social interaction metrics for Mozilla FIREFOX 3.0

$\log(Y_i)$	MB		M1		M2		M3		M4		M5	
CHURN	**0.649**	⋆	**0.663**	⋆	**0.676**	⋆	**0.707**	⋆	**0.726**	⋆	**0.603**	⋆
NSOURCE			**−0.050**	○	−0.044		−0.026		0.008		−0.039	
NSCOM			−0.100		−0.019		0.080		0.112		**0.204**	○
PATCHS			0.697		0.929		0.790		**0.838**	○	1.632	
NLINK			**0.078**	•	**0.139**	⋆	**0.178**	⋆	**0.169**	⋆	**0.098**	•
NPART					**−0.752**	⋆	−0.169		0.031		−0.164	
NUSERS					**0.214**	⋆	**0.286**	⋆	**0.300**	⋆	**0.198**	•
SNACENT					**3.082**	•	1.646		0.312		1.827	
REPLY							**−0.032**	○	−0.006		0.000	
REPLYE							**−1.803**	⋆	**−1.172**	⋆	−0.608	
DLEN							0.011		**0.048**	○	0.018	
DLENE							−0.089		**0.746**	•	0.061	
INT							**−0.295**	⋆	**−0.254**	⋆	−0.073	
INTE							**0.486**	•	**0.599**	•	0.453	
WA									**−0.308**	⋆	**−0.146**	⋆
WAE									0.291		0.628	
CON1−3												⋆
Chi−Sq	**744.14**	⋆	**773.88**	⋆	**807.54**	⋆	**933.29**	⋆	**1007.11**	⋆	**1836.43**	⋆
Dev.Expl	14.90%		15.49%		16.17%		18.68%		20.16%		36.76%	
Delta Chisq			29.74		33.66		125.75		73.82		829.32	

⋆ $p < 0.001$, • $p < 0.01$, ○ $p < 0.05$, + $p < 0.1$

References

Alatis JE (1993) Language, communication and social meaning. Georgetown University Press

Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc Y-G (2008) Is it a bug or an enhancement?: a text-based approach to classify change requests. In: CASCON '08: proceedings of the 2008 conference of the center for advanced studies on collaborative research. ACM, pp 304–318

Anvik J, Hiew L, Murphy GC (2006) Who should fix this bug? In: ICSE '06: proceedings of the 28th international conference on software engineering. ACM, pp 361–370

Bacchelli A, D'Ambros M, Lanza M (2010) Are popular classes more defect prone? In: To appear in FASE 2010: proceedings of the 13th international conference on fundamental approaches to soft. eng. Springer

Basili VR, Briand LC, Melo WL (1996) A validation of object-oriented design metrics as quality indicators. IEEE Trans Softw Eng 22(10):751–761

Bettenburg N, Just S, Schröter A, Weiss C, Premraj R, Zimmermann T (2008) What makes a good bug report? In: SIGSOFT '08/FSE-16: proceedings of the 2008 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 308–318

Bettenburg N, Premraj R, Zimmermann T, Kim S (2008) Extracting structural information from bug reports. In: MSR '08: proceedings of the 2008 international working conference on mining software repositories. ACM, pp 27–30

Bird C, Pattison D, D'Souza R, Filkov V, Devanbu P (2008) Latent social structure in open source projects. In: ESEC/FSE '08: proceedings of the 2008 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 24–35

Bird C, Bachmann A, Aune E, Duffy J, Bernstein A, Filkov V, Devanbu P (2009) Fair and balanced?: bias in bug-fix datasets. In: ESEC/FSE '09: proceedings of the 2009 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 121–130

Bland MJ, Altman DG (1996) Transformations, means and confidence intervals. Br Med J 312(7038):1079

Cataldo M, Mockus A, Roberts JA, Herbsleb JD (2009) Software dependencies, work dependencies, and their impact on failures. IEEE Trans Softw Eng 35(6):864–878

Cohen J (2003) Applied multiple regression/correlation analysis for the behavioral sciences, vol 1. Routledge

Čubranić D, Murphy GC (2003) Hipikat: recommending pertinent software development artifacts. In: ICSE '03: proceedings of the 25th international conference on software engineering. IEEE Computer Society, pp 408–418

D'Este C (2004) Sharing meaning with machines. In: Proceedings of the fourth international workshop on epigenetic robotics. Lund University Cognitive Studies, pp 111–114

Edwards AWF (1963) The measure of association in a 2 by 2 table. J R Stat Soc A 126(1):109–114

Fay MP, Proschan MA (2010) Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. In: Statistics surveys, vol 4, pp 1–39

Fischer M, Pinzger M, Gall H (2003) Analyzing and relating bug report data for feature tracking. In: WCRE '03: proceedings of the 10th working conference on reverse engineering. IEEE Computer Society, p 90

Friendly M (2002) Corrgrams: exploratory displays for correlation matrices. Am Stat 56(1):316–324

Guo PJ, Zimmermann T, Nagappan N, Murphy B (2010) Characterizing and predicting which bugs get fixed: an empirical study of microsoft windows. In: To appear in proceedings of the 32th international conference on software engineering

Guo PJ, Zimmermann T, Nagappan N, Murphy B (2011) Not my bug and other reasons for software bug report reassignments. In: Proceedings of the ACM conference on computer supported cooperative work (CSCW 2011). ACM

Hassan AE (2009) Predicting faults using the complexity of code changes. In: ICSE '09: proceedings of the 31st international conference on software engineering. IEEE Computer Society, pp 78–88

Jeong G, Kim S, Zimmermann Th (2009) Improving bug triage with bug tossing graphs. In: ESEC/FSE '09: proceedings of the 2009 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 111–120

Kutner MH, Nachtsheim CJ, Neter J (2004) Applied linear regression models, 4th international edn. McGraw-Hill/Irwin

McCabe TJ (1976) A complexity measure. In: ICSE '76: proceedings of the 2nd international conference on software engineering. IEEE Computer Society Press, p 407

Meneely A, Williams L, Snipes W, Osborne J (2008) Predicting failures with developer networks and social network analysis. In: SIGSOFT '08/FSE-16: proceedings of the 2008 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 13–23

Mertsalov K, Magdon-Ismail M, Goldberg M (2009) Models of communication dynamics for simulation of information diffusion. In: Proceedings of the 2009 international conference on advances in social network analysis and mining (ASONAM '09). IEEE Computer Society Press, pp 194–199

Mockus A, Zhang P, Li PL (2005) Predictors of customer perceived software quality. In: ICSE '05: proceedings of the 27th international conference on software engineering. ACM, pp 225–233

Mockus A, Nagappan N, Dinh-Trong T (2009) Test coverage and post-verification defects: a multiple case study. In: Proceedings of the 2009 3rd international symposium on empirical software engineering and measurement (ESEM '09). IEEE Computer Society, pp 291–301

Munson JC, Elbaum SG (1998) Code churn: a measure for estimating the impact of code change. In: ICSM '98: proceedings of the international conference on software maintenance. IEEE Computer Society, p 24

Nagappan N, Ball T (2005) Use of relative code churn measures to predict system defect density. In: ICSE '05: proceedings of the 27th international conference on software engineering. ACM, pp 284–292

Nagappan N, Ball T (2007) Using software dependencies and churn metrics to predict field failures: an empirical case study. In: ESEM '07: proceedings of the first international symposium on empirical software engineering and measurement. IEEE Computer Society, pp 364–373

Ohlsson N, Alberg H (1996) Predicting fault-prone software modules in telephone switches. IEEE Trans Softw Eng 22(12):886–894

Purao S, Vaishnavi V (2003) Product metrics for object-oriented systems. ACM Comput Surv 35(2):191–221

Pinzger M, Nagappan N, Murphy B (2008) Can developer-module networks predict failures? In: SIGSOFT '08/FSE-16: proceedings of the 2008 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 2–12

Shannon CE (2001) A mathematical theory of communication. SIGMOBILE Mob Comput Commun Rev 5(1):3–55

Schröter A, Zimmermann T, Zeller A (2006) Predicting component failures at design time. In: ISESE '06: proceedings of the 2006 ACM/IEEE international symposium on empirical software engineering. ACM, pp 18–27

Schroter A, Bettenburg N, Premraj R (2010) Do stack traces help developers fix bugs? In: Proceedings of 7th IEEE working conference on mining software repositories (MSR'10). IEEE Computer Society, pp 118–121

Shihab E, Ihara A, Kamei Y, Ibrahim WM, Ohira M, Adams B, Hassan AE, Matsumoto K (2010a) Predicting re-opened bugs: a case study on the eclipse project. In: Proceedings of the 17th working conference on reverse engineering (WCRE 2010). IEEE Computer Society, pp 13–16

Shihab E, Jiang ZM, Ibrahim WM, Adams B, Hassan AE (2010b) Understanding the impact of code and process metrics on post-release defects: a case study on the eclipse project. In: Proceedings of the 4th IEEE international symposium on empirical software engineering and measurement (ESEM 2010). ACM, pp 4:1–4:10

Śliwerski J, Zimmermann T, Zeller A (2005) When do changes induce fixes? In: MSR '05: proceedings of the 2005 international workshop on mining software repositories. ACM, pp 1–5

Steel RGD, Torrie JH (1960) Principles and procedures of statistics. McGraw-Hill, pp 187–287

Wasserman S, Faust K (1994) Social network analysis: methods and applications (structural analysis in the social sciences), 1st edn. Cambridge University Press

Wolf T, Schröter A, Damian D, Nguyen T (2009) Predicting build failures using social network analysis on developer communication. In: ICSE '09: proceedings of the 31st international conference on software engineering. IEEE Computer Society, pp 1–11

Yin RK (1994) Case study research: design and methods. Sage, Thousand Oaks, California

Zeller A (2009) Why programs fail, 2nd edn: a guide to systematic debugging. Morgan Kaufmann

Zimmermann T, Nagappan N (2008) Predicting defects using network analysis on dependency graphs. In: ICSE '08: proceedings of the 30th international conference on software engineering. ACM, pp 531–540

Zimmermann T, Nagappan N, Gall H, Giger E, Murphy B (2009) Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: ESEC/FSE '09: proceedings of the 2009 ACM SIGSOFT symposium on foundations of software engineering. ACM, pp 91–100

Zimmermann T, Premraj R, Zeller A (2007) Predicting defects for eclipse. In: Proceedings of the third international workshop on predictor models in software engineering, May 2007

Nicolas Bettenburg is a Ph.D. Candidate under the supervision of Ahmed E. Hassan at Queen's University. He received the B.Sc. and M.Sc. degrees in computer science from Saarland University in 2006 and 2008, respectively.

His research interests focus around mining unstructured information from software repositories with a focus on understanding the relationship between developer collaboration and software quality. His professional activities include the co-organization the 2010 Workshop on Mining Unstructured Data (MUD), and carrying out peer-reviews for a variety of journals in the research area of Empirical Software Engineering. He is a member of the IEEE and ACM.

Ahmed E. Hassan is the NSERC/RIM Industrial Research Chair in Software Engineering for Ultra Large Scale systems at Queen's University. Dr. Hassan spearheaded the organization and creation of the Mining Software Repositories (MSR) conference and its research community. He co-edited special issues of the IEEE Transactions on Software Engineering and the Journal of Empirical Software Engineering on the MSR topic.

Early tools and techniques developed by Dr. Hassan's team are already integrated into products used by millions of users worldwide. Dr. Hassan industrial experience includes helping architect the Blackberry wireless platform at RIM, and working for IBM Research at the Almaden Research Lab and the Computer Research Lab at Nortel Networks. Dr. Hassan is the named inventor of patents at several jurisdictions around the world including the United States, Europe, India, Canada, and Japan.