Studying the Impact of Social Structures on Software Quality

Nicolas Bettenburg and Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
School of Computing, Queen's University
Kingston, Ontario, Canada
Email: {nicbet, ahmed}@cs.queensu.ca

***Abstract*—Correcting software defects accounts for a significant amount of resources such as time, money and personnel. To be able to focus testing efforts where needed the most, researchers have studied statistical models to predict in which parts of a software 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 file-oriented measures, such as source code and churn metrics. 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 repositories used to manage development efforts. In this paper, we investigate statistical models, to study the impact of social structures between developers and end-users on software quality. These models use predictor variables based on social information mined from the issue tracking and version control repositories of a large open-source software project. The results of our case study are promising and indicate that statistical models based on social information have a similar degree of explanatory power as traditional models. Furthermore, our findings suggest that social information does not substitute, but rather augments traditional product and process-based metrics used in defect prediction models.**

***Keywords*-Human Factors; Software Evolution; Metrics/Measurement; Software Quality Assurance;**

I. Introduction

In the foreword to "Why programs fail" [31], James Larus, director of Microsoft's 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 concept that sparks enormous research efforts. Software contains errors, and fixing these errors is very costly – even more so, if they are discovered after the software has shipped.

To keep these costs at minimum, researchers have long-since studied two core areas in empirical software engineering: understanding and minimizing the cause of errors and building effective systems to predict where these errors are likely to occur in the software [8]. Both research areas are intertwined, as knowledge gained from understanding root causes can help in building better predictors [25]. At the same time the study of prediction models provides cues for understanding the causes of errors, such as complex code change processes [16]. Past work in defect prediction makes extensive use of product and process measures that can be obtained from the source code of a software, such as code complexity metrics [18], change metrics [21] and inter-dependencies of elements in the code [23].

However, source-code is the end product of a variety of collaborative activities carried out by the developers of a software. Lately, researchers begin to understand that the intricacies of these activities such as social networks [30], work dependencies [10] and daily routines [27] stand in relation to 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, email communication archives. In this study we investigate how we can use information about the social interactions between developers and users in defect prediction and set out to study their impact on software quality. We use statistical models to establish and inspect mathematical dependencies between these measurements – an approach that been successfully used in previous research to study the relation between measures of source code and defects [10], [16], [22], [23], [25], [34].

In particular, we set out to study the following relations between measures of social interaction and software quality.

(1) The relationship between the social structures formed by developers and users, and software quality.
(2) The relationship between the contents of communication that takes place between developers and users, and software quality.
(3) The relationship between the dynamics of this communication and software quality.

Through a case study on the `ECLIPSE` software system, we find that such relationships exist and that they can be used to create prediction models with explanatory power similar to models based on code metrics. In addition, we find that a combination of social interaction measures and complexity metrics yields higher explanatory power than each of the models taken separately.

The rest of this paper is organized as follows. We discuss work that is closely related to our study in Section II. In Section III we present the data we base our case study on

and describe the set of information measures we calculate from this data in Section IV. The discussion of our study design in Section V is then followed by a presentation of our case study. We close the paper by discussing possible threats to the validity of the work (Section VI), and our conclusion (Section VII).

II. Related Work

In the following we discuss related research from the two major research areas of defect prediction and social analyses of software development.

A. Defect Prediction

Several researchers have previously investigated the use of data captured from version control systems and bug databases for defect prediction. Basili et al. [5] 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 [24]. Extensive work by Nagappan et al. [22], [23] 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 [25]. Hassan demonstrates in a large case study that prediction models based on change complexity outperform traditional churn based prediction models [16]. Zimmermann et al. use social network measures on dependency graphs to predict defects [32]

In contrast to previous studies, the work presented in this paper does not focus on formulating accurate prediction models, but rather 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. We use prediction models as an explorative tool in the same vein of work done by Mockus et al. [19], [20].

B. Social Analysis

The research work probably most related to ours is a study by Wolf et al. [30], which 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.

A similar study was conducted by Bacchelli et al. [4] 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.

III. Data Collection

For our case study, we used two main sources of data available for the ECLIPSE project. 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 [2] and for the remainder of this paper we will refer to them as "issue reports". Every report contains a variety of supporting meta-information such as a unique identification number, the software version and operating system it relates to, 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 this study we do not need to distinguish the initial description from the following discussion and hence treat the description as the first discussion message. 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 software 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 [27]. 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. Links start out with a low trust at first, which 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 [13], [25], [27], [28] with high success. To further increase the quality of links, we incorporated the improvements by Bird et al. [8]. Through these links we can then associate issue reports with files. Overall, we were able to establish 67,705 such links.

IV. Social Interaction Measures

In this section we describe the social interaction measures that we use in our statistical models. For each measure, we briefly motivate its inclusion and outline our approach to measure it. Our social measures are determined from traces of activity that developers leave behind in the issue tracking system. Hence we calculate each measure on a per-issue

report level. However, as we will later study the relation between social interaction measures and software quality on a per-file level, we need to aggregate the measures across all issue reports associated with a file. Our default method of aggregation is to take the *average*. However, for some measures we are interested in their variability and for these measures we will use *entropy* for aggregation. Entropy is a concept we borrow from information theory [26]. The normalized entropy is defined as

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

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 [26] and allows us to compare entropy measures across different distributions.

A. Measures of Discussion Contents

In a previous study [6], 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 a problem is crucial for addressing a reported issue adequately, we conjecture that the presence or absence of such 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 implement regression tests that safeguard the software against an accidental reintroduction of the problem in the future. Therefore, we choose to incorporate structural elements as factors in our statistical models. We used the infoZilla tool [7] to extract structural elements from the textual contents of bug report discussions. In the following we describe the seven measures of discussion contents we used in our study.

1) Source Code: Our first measure 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. 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 *source code complexity* (NSCOM) as a qualitative measure for each of the source code examples. Since the discussions often contain a mix of natural language text and structural elements, we use McCabe's cyclomatic complexity [18], rather than lines of code as our complexity measure.

2) Patches: Our second measure is the *amount of patches* (NPATCH) provided in the discussions. Publicly discussed patches provide peer-reviewed solutions to the reported issues and as such are less likely to contain errors. In addition to this quantitative measure we also compute a qualitative measure of patches by recording the *number of files changed by a patch* (PATCHS), which measures how spread out changes provided by the patch are. 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 correctly fix the reported issues).

3) Stack Traces: Our third measure records the *amount of stack traces* (NTRACE) provided in the contents. Information inside stack traces provide 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 [31]. We use the number of methods reported in the stack traces as a qualitative measure for the *size of stack traces* (TRACES).

4) Links: Our fourth measure 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.

B. Measures of Social Structures

In addition to the information obtained from the textual contents of discussions, we also compute a number of measures to describe the social structures created through issue reports. In the following we describe the five measures of social structures used in our study.

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 his activity in the system. We conjecture that the total amount of unique participants in the discussion of an issue report is an indicator for the relative importance of the reported problem. Our first measure hence counts the *number of unique participants* (NPART) in the discussion.

2) Role: In this study we further categorize participants into two different roles: developers and users. We consider a participant as a developer if he was assigned to fixing at least one issue reported 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).

3) Reputation: Another social property of participants, which is orthogonal to their role is their degree of reputation in the community. In our study, we determine the top three

participants with the highest reputation (expressed by the past amount of contributed messages) for each discussion attached to issue reports. These measures are captured in the three factor variables (CON1), (CON2), and (CON3). The degree of reputation of a participant can influence the development process connected to an issue; for example Guo et al. show that defects reported by more reputable users have a higher likelihood to get fixed [15]. As each factor variable introduces many degrees of freedom to the model, we limit our study to the top three most reputable developers.

4) Centrality: Our last measure of social structure is taken from the area of social network analysis. For each discussion attached to an issue report in the bug database we first construct a discussion flow graph. 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 the message senders. We express the interconnectedness of nodes (participants) in the discussion flow graph as a measure of *closeness-centrality* (SNACENT). This measure focuses on how close each participant is to all other participants in the discussion [29]. However, closeness-centrality is a per-node measure, yet we want to express whether there is a healthy discussion between all participants, or whether there is a smaller set of key participants that drive the communication back and forth. To aggregate the closeness-centrality of all participants in the discussion into a single value, we use the normalized entropy measure. We need this normalization, as discussions do not all have the same amount participants.

C. Measures of Communication Dynamics

In addition to information about the discussion content and the actors involved, we can measure the dynamics of a discussion. In the following we describe the six measures of communication dynamics we used in our study.

1) Number of Messages: By their very nature, issues that are complex, not well understood, or controversial require a greater amount of communication than simple problems. We represent this idea by a measure of the *amount of messages* (NMSG) exchanged in a discussion.

2) Length of Messages: In addition to this quantitative measure, we define two qualitative measures: first, the *number of words in a discussion* (DLEN), and second *discussion length entropy* (DLENE). 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.

3) Reply Time: Cognitive sciences define communication as "the sharing of meaning" [1], [11]. The absence of communication for an extended period of time, or distorted communication, can be the cause of 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 the *mean reply time between messages* (REPLY), and the *reply time entropy* (REPLYE) for discussions.

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 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 issue report discussion. In addition we capture the variability of interestingness in a measure of *interestingness entropy* (INTE).

D. Workflow Measures

Issue reports represent work items for developers and follow a set of states from creation until closure [3] 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, 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 *variability of workflow activity* (WAE).

V. Analysis and Results

Our analysis uses statistical models to investigate the relation between social information and software quality. We do so by exploring the statistical relations between failure proneness of files and the measures of social information 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 measures and classical code-metrics based models.

A. Study Design

Following previous work in defect prediction [23], 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 measures described in Section III 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. In order to later compare our statistical models to previous work, we choose to perform our measures for periods of 6 months surrounding the release of `Eclipse 3.0`. From the measurements obtained for this time period, we create linear regression models that set

the amount of *post-release defects* into relation of our *pre-release measures*. The complete linear regression model has the form

$$\begin{aligned} Defects = & \sum_i \alpha_i \cdot ContentMeasure_i \\ & + \sum_j \beta_j \cdot StructuralMeasure_j \\ & + \sum_k \gamma_k \cdot ComDynMeasure_k \\ & + \sum_l \delta_l \cdot WorkflowMeasure_l + \epsilon \end{aligned}$$

Based on this model, we will investigate the statistical relationships between the social interaction measures, which are represented by the *regression variables* in the model, and post release defects, represented by 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 measurements we collected. 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 that each of the four dimensions of social measures has on post-release defects. Our approach is similar to the work by Cataldo and Mockus [10], [20].

We follow a hierarchical modelling approach when creating these models: we start out with a baseline model that uses classical defect predictors as regression variables. We then build subsequent models to which we step-by-step add our content, structure, communication dynamics and workflow measures, and report for each statistical model the explanatory power, χ^2, of the model and the percentage of deviance explained. The deviance for each model M_i is defined as $D(M_i) = -2 \cdot LL(M_i)$, 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 $D(M_i)$. In addition we test for each subsequent model, whether the difference from the model it is derived from is statistically significant and present the corresponding p-level. A hierarchical modelling approach has the advantage over a step-wise modelling approach that it minimizes artificial inflation of errors and thus overfitting.

B. Preliminary Analysis of Social Interaction Measures

Our four groups of social interaction measures (content, structure, dynamics, workflow) represent different characteristics of collaborative development on work items that are represented by issue reports. While content measures are more explicit in capturing the information exchanged between developers and users, our measures of social structure are more implicit and capture the latent relationships and roles of stakeholders. Table I presents a summary of our measures 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 [9]. Figure 1 summarizes the pairwise correlations between our 20 regression variables and our dependent variable in a correlogram visualization [14]. A correlogram reports for each unique pair of variables the strength of the correlation as a colour-coded field (red for positive 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.

	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

Table I
DESCRIPTIVE STATISTICS OF SOCIAL INTERACTION MEASURES

We identify the following types of intercorrelations in our dataset that could pose problems in our statistical modelling. First, we observe correlations between measures from different concepts. 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). We believe the first of these three intercorrelations stem from a default setting in the issue tracking systems that puts contributors automatically on the notification lists, but can not offer an explanation for the correlation between the URLs provided by users in the discussion contents and interestingness.

Second, we observe correlations through redundancy. We expected such correlations when designing social interaction measures: naturally 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).

The third type of observed hotspots are moderate correlations between quantitative measures and qualitative measures, e.g., between the number of patches and the average

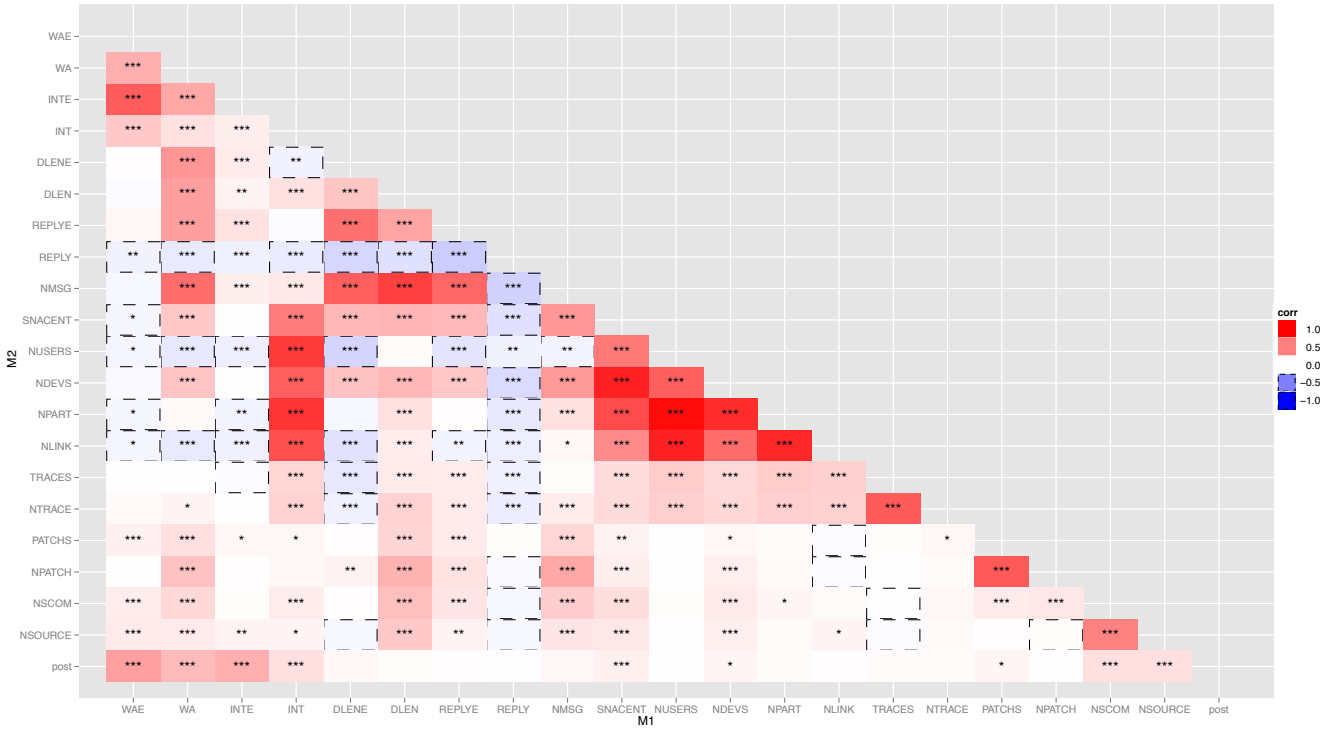

Figure 1. Pairwise correlations of social interaction measures with levels * p<0.05, ** p<0.01, *** p<0.001. Strength of correlations is indicated by colour intensities; negative correlations are marked with a dashed outline.

	Variance Inflation Factor		
log(Y_i)	**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

Table II
STEP-WISE ANALYSIS OF MULTICOLLINEARITY.

size of patches.

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. To investigate potential problems, we compute the variance inflation factors for each variable. Variance inflation factors are widely used to measure the degree of multi-collinearity between variables in regression models [17]. Following Kutner et al. [17], we 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 variance inflation factors for this model are presented in Table II, Model 1. We observe two variables that have a variance inflation factors greater than 10. We remove the highest one (NMSG) from the regression model and recompute the variance inflation factors with the reduced set of variables. The resulting model, (Model 2 in Table II) contains only one more variable with an inflation factor 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 II), no variables have an inflation factor larger than 5 and we finish our analysis of multicollinearity.

C. Hierarchical Analysis Design

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 measures on the post-release defects.

The results of our hierarchical analysis are presented in Table III. To make the interpretation of the coefficients of the regression variables easier, we report the odds ratios [12] of each measure, rather than the coefficients themselves. An odds ratio greater than one indicates a positive relation between the dependent variable (post-release defects) and the independent variables (social interaction measures), whereas an odds ratio smaller than one indicates a negative relation. As we are working in a log-transformed space, the

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. 2**	***
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	

*** p<0.001, ** p<0.01, * p<0.05, + p <0.1

Table III
HIERARCHICAL ANALYSIS OF LOGISTIC REGRESSION MODELS ALONG THE FOUR DIMENSIONS OF SOCIAL INTERACTION MEASURES.

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 untransformed space.

We start our hierarchical analysis with a *baseline model* which relates churn [21] to post-release defects. Churn has been shown in the past to be a valuable code-based predictor of defects [22], [23] even when used across projects [33]. 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 III 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 [22], [23].

Model M1 introduces the first dimension of social interaction we want to study: the measures of structural information items in the contents of issue report discussions. The results of the logistic regression model show, that only specific structural information items are statistically significant. Whereas the number of source code examples, the number of links and the effect size of patches stay significant throughout all models when new variables are introduced, the total number of patches plays only a marginal role, indicating that this measure is unlikely to impact future failure proneness. When looking at the odds ratios, we surprisingly find a positive link between the number of source code samples and future defects, as we initially expected code samples to have a beneficial effect. One possible explanation might be that developers trust user provided sample solutions and incorporate their proposed (yet possibly flawed) modifications without further verification. The second strongest relationship observed from the results is the positive relation between the number of links provided by users and failure proneness. The third relationship that we observe is a very high, positive relation between the effect size of patches and future failure proneness of files. This result confirms earlier findings on the risk of scattered changes [16]. *Overall the results show that measures of structural elements in issue report discussions 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 change is statistically significant.*

Model M2 introduces the second dimension of social interaction measures: structural information. The results show that the role of a 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 reputation from this model, as we record them as factors with many levels that may disrupt our hierarchical modelling approach. We will revisit these measures later in model M5. *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 measures form the category of communication dynamics. The results show a statistically significant and strongly negative relation between the mea-

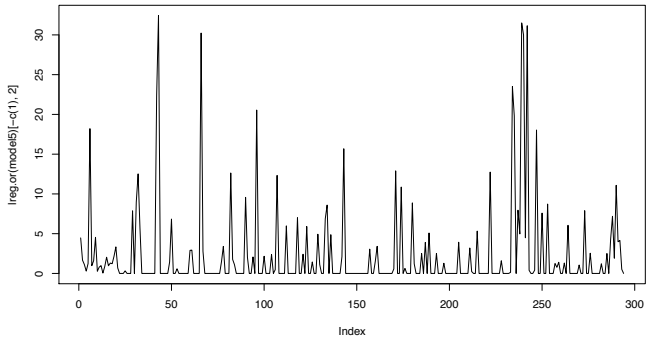

Figure 2. Odds ratios for reputation measures in model M5, each index represents one distinct level (developer) of the factor variables.

sure of reply time entropy and future failure likelihood, yet not 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 reputation in model M5. *Overall, we observe a strong effect of inconsistencies in discussion flow on the future 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 measures used in our study: workflow activity. Our results show a strong positive relation between the total amount of workflow activities and post-release defects. However this relation is rendered less significant when we later introduce measures of reputation in model M5. *Our findings suggest that workflow activities play only a marginal, complementary role in the relation between social interaction measures 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 measures.*

We revisit the dimension of social structures in Model M5, by adding our measures of reputation. These measures are expressed as three factors (with discussion participants as levels) and measure the top3 reputable contributors for each discussion. As each factor generates as many binary dummy variables as its levels we do not show the complete model. However, we measure the inherent effect of the factors on the statistical model with type II ANOVA tests and present a plot of the odds ratios of each factor level in Figure 2. Our analysis of variance tests for the reputation measures show that they are statistically significant at $p < 0.001$. From the plot of odds ratios we observe a relation between the presence of certain reputable participants in a discussion and post-release defects, indicated by distinct spikes in the plot. One possible explanation for this relation might be that specific participants are responsible for certain areas of a project's code. These experts contribute the most to discussion on this area and become associated with defects. However, when introducing these factors our focus lies less on the relationship between the presence of specific top reputable contributors, but rather on extending the explanatory power of the model. 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. *Overall, the introduction of reputation measurements increases the explanatory power of the extended model significantly. The increase of 11.66% is statistically significant. The large increase shows that the presence of particular contributors in a discussion can act as a valuable indicator for future failure proneness.*

	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 IV
BASELINE MODEL M

D. Enhancing Traditional Models with Social Information

In the final part of our analysis we want to investigate whether social information measures 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 [34]. As Bird et al. note [8], this dataset is extensively documented and has been widely used in research. Among other informations, 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. We first re-create the original code-metrics based model created by Zimmermann et al. [34]. The original statistical model M is presented in Table IV. 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' using the set of social interaction measures that we found statistically significant in our previous analysis and compare it to the original model M. The extended model is presented in Table V. The addition of social interaction measures increases the explanatory power of model M by $\chi^2 = 716.55$ to a

	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
CON1-CON3	—	—	—	—

Table V
AUGMENTED MODEL M'

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%. The observed increase is statistically significant at $p < 0.001$. *The large increase in explanatory power of the augmented model demonstrates that social interaction measures are valuable for complementing traditional prediction models based on product and process measures.*

VI. CONSIDERATIONS OF VALIDITY

We now discuss potential threats to the validity of our study. The first concern is generalizability. In our analysis we presented a single case study surrounding a major release of a large open-source software system. However based on this limited scope, our results might not generalize to other projects and domains. Second, our analysis of the relation between social interaction measures and post-release defects can not claim causal effects, as we are investigating correlations, rather than conducting impact studies. For example, even though we have seen relation between workflow activities and defects, it could be the case that increased workflow activity is generated because a file contains many defects. Third, recent research has raised numerous concerns about the quality of data contained in version archives [2], [8]. To account for major source of bias we performed a careful selection of features and avoided to base our social interaction measures on features of issue reports that were reported to be highly biased. Fourth, the selection of measurements might not be complete: our measurements might capture only the symptomatic effects of other variables. We tried to counter this threat by performing a selection of measurements along four different dimensions and conducting an analysis of intercorrelation and multicollinearity using variance inflation factors.

VII. CONCLUSION

In this study we investigated the impact of social interaction measures on software quality, expressed through their impact on post-release defects. We observed that the presence of certain information items in the contents of discussions were considerably more relevant than of roles of participants, in relation to the future failure proneness of the software. Similarly we discovered that the consistency of the communication flow stands in strong relation with failure proneness. In summary we observe the following relationships from Table III.

- For every unit increase in logarithmic space of changes made to a file, then chances of failure proneness increase 4.5 times.
- For every log-unit increase of source code items discussed, the failure proneness increases by 67%.
- For every log-unit increase in the number of files changed by a discussed patch, the odds of failure-proneness increase 18 times.
- A consistent information flow in discussions decreases the likelihood of future defects significantly.
- The participation of particular contributors is a significant indicator for future failure-proneness.

These observations open a variety of research opportunities in order to investigate their intrinsic effects more deeply. In addition to demonstrating a connection between social activities in development and software defects we have shown that social information can not only explain a similar amount of variance than traditional models but complements traditional complexity-based defect prediction models increasing their explanatory power. In future work, we plan to explore these relationships in more detail and further investigate the possibilities of defect prediction based on social metrics.

ACKNOWLEDGEMENT

We want to thank Audris Mockus of Avaya Labs for his helpful comments on early versions of this study.

REFERENCES

[1] J. E. Alatis, *Language, communication and social meaning*. Georgetown University Press, 1993.

[2] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y.-G. Guéhéneuc, "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, 2008, pp. 304–318.

[3] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *ICSE '06: Proceedings of the 28th International Conference on Software Engineering*. ACM, 2006, pp. 361–370.

[4] A. Bacchelli, M. D'Ambros, and M. Lanza, "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, 2010.

[5] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751–761, 1996.

[6] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in *SIGSOFT '08/FSE-16: Proceedings of the 2008 ACM SIGSOFT Symposium on Foundations of Software Engineering*. ACM, 2008, pp. 308–318.

[7] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Extracting structural information from bug reports," in *MSR '08: Proceedings of the 2008 International Working Conference on Mining Software Repositories*. ACM, 2008, pp. 27–30.

[8] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "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, 2009, pp. 121–130.

[9] M. J. Bland and D. G. Altman, "Transformations, means and confidence intervals." *British Medical Journal*, vol. 312, no. 7038, p. 1079, 1996.

[10] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp. 864–878, 2009.

[11] C. D'Este, "Sharing meaning with machines," in *Proceedings of the Fourth International Workshop on Epigenetic Robotics*. Lund University Cognitive Studies, 2004, pp. 111–114.

[12] A. W. F. Edwards, "The measure of association in a 2 by 2 table," *Journal of the Royal Statistical Society. Series A (General)*, vol. 126, no. 1, pp. 109–114, 1963.

[13] M. Fischer, M. Pinzger, and H. Gall, "Analyzing and relating bug report data for feature tracking," in *WCRE '03: Proceedings of the 10th Working Conference on Reverse Engineering*. IEEE Computer Society, 2003, p. 90.

[14] M. Friendly, "Corrgrams: Exploratory displays for correlation matrices," *The American Statistician*, vol. 56, no. 1, pp. 316–324, November 2002.

[15] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "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*, 2010.

[16] A. E. Hassan, "Predicting faults using the complexity of code changes," in *ICSE '09: Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 2009, pp. 78–88.

[17] M. H. Kutner, C. J. Nachtsheim, and J. Neter, *Applied Linear Regression Models*, fourth international ed. McGraw-Hill/Irwin, September 2004.

[18] T. J. McCabe, "A complexity measure," in *ICSE '76: Proceedings of the 2nd International Conference on Software Engineering*. IEEE Computer Society Press, 1976, p. 407.

[19] A. Mockus, N. Nagappan, and T. T. Dinh-Trong, "Test coverage and post-verification defects: A multiple case study," in *ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 291–301.

[20] A. Mockus, P. Zhang, and P. L. Li, "Predictors of customer perceived software quality," in *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*. ACM, 2005, pp. 225–233.

[21] J. C. Munson and S. G. Elbaum, "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, 1998, p. 24.

[22] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," in *ICSE '05: Proceedings of the 27th International Conference on Software Engineering*. ACM, 2005, pp. 284–292.

[23] ——, "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, 2007, pp. 364–373.

[24] N. Ohlsson and H. Alberg, "Predicting fault-prone software modules in telephone switches," *IEEE Trans. Softw. Eng.*, vol. 22, no. 12, pp. 886–894, 1996.

[25] A. Schröter, T. Zimmermann, and A. Zeller, "Predicting component failures at design time," in *ISESE '06: Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*. ACM, 2006, pp. 18–27.

[26] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, 2001.

[27] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" in *MSR '05: Proceedings of the 2005 International Workshop on Mining Software Repositories*. ACM, 2005, pp. 1–5.

[28] D. Čubranić and G. C. Murphy, "Hipikat: recommending pertinent software development artifacts," in *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*. IEEE Computer Society, 2003, pp. 408–418.

[29] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*, 1st ed. Cambridge University Press, 1994.

[30] T. Wolf, A. Schröter, D. Damian, and T. Nguyen, "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, 2009, pp. 1–11.

[31] A. Zeller, *Why Programs Fail, Second Edition: A Guide to Systematic Debugging*. Morgan Kaufmann, 2009.

[32] T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *ICSE '08: Proceedings of the 30th international conference on Software engineering*. ACM, 2008, pp. 531–540.

[33] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "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, 2009, pp. 91–100.

[34] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, May 2007.