Electronic Communications of the EASST
Volume 65 (2014)

Proceedings of the International Workshop on Software Quality and Maintainability (SQM 2014)

Comparing communication and development networks for predicting file change proneness: An exploratory study considering process and social metrics

Igor Scaliante Wiese, Douglas Nassif Junior, Reginaldo Re, Igor Steinmacher, Marco Aurelio Gerosa

15 pages

Guest Editors: Lodewijk Bergmans, Tom Mens, Steven Raemaekers
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Comparing communication and development networks for predicting file change proneness: An exploratory study considering process and social metrics

Igor Scaliante Wiese[1], Douglas Nassif Junior[2], Reginaldo Re[3], Igor Steinmacher[4], Marco Aurelio Gerosa[5]

[1] igor@utfpr.edu.br, [2] nassifrroma@gmail.com, [3] reginaldo@utfpr.edu.br, [4] igorfs@utfpr.edu.br
Department of Computing - Federal University of Technology - Parana (UTFPR)
Campo Mourao, PR, Brazil
[5] gerosa@ime.usp.br http://lapessc.ime.usp.br/
Department of Computer Science - University of Sao Paulo (USP)
Sao Paulo, SP, Brazil

Abstract: Previous studies have shown that social factors of software engineering influence software quality. Communication and development networks represent the interactions among software developers. We explored the statistical relationships between file change proneness and a set metrics extracted from the issue tracker and version control system data to find the relative importance of each metric in understanding the evolution of file changes in the Rails project. Using hierarchical analysis, we found that code churn, number of past changes, and number of developers explain the evolution of changes in the Rails project better than Social Network Analysis (SNA) metrics. Considering the relative importance of each predictor, we got the same results. We also conducted a factor analysis and found that social metrics contribute to explain a group of files different from those explained by process metrics.

Keywords: Social Network, Social Metrics, Evolution of changes, Prediction Change Proneneness, Github

1 Introduction

Prediction models enable the identication of files that change more often during software evolution. These files may require more attention from the developers [Sch00]. Researchers have been using process and social metrics to build their prediction models. Previous studies have shown that social factors in software development influence software quality [BNG+, SHAJ12, BH13]. Mailing lists, issue/bug trackers and version control systems are some of the most common tools used as part of the infrastructure to support interaction among developers. The data extracted from the history of these tools can be used to better understand the change proneness of artifacts in software development.

Developers cooperate asynchronously, changing files during software evolution. Understanding the structure of this cooperation may tell us a lot about the quality of the software [MWSO08].

Moser et al.[MPS08], for example, used code change metrics such as code churn, number of developers, and number of past changes to build prediction models. Their findings show that historical data about le changes contain discriminatory and meaningful information about software quality [MPS08]. Several types of socio-technical networks were used to build prediction models [MWSO08, WSDN09, BNG+, BBC11], but they present controversial results [HBB+12]. For example, Bird et al. [BNG+] report better predictive performance when social metrics are used as an element within a socio-technical network. On the other hand, Ostrand et al. [WOB07] report that social metrics do not improve predictive performance.

Considering these controversial results regarding social metrics, we want to explore different types of social networks to compare process and social metrics in consideration of file changes. We did not consider source code metrics, as we were more interested in collaborative aspects of software engineering. Besides, Rahman and Devanbu [RD13] showed that process metrics lead to better results than source code. We investigated two research questions: RQ1: What are the differences when process and social metrics related to communication and cooperation networks are used to predict file change proneness? and RQ2: What is the relative importance of each metric to predict file change proneness?

To conduct this study, we gathered data from three semesters of the Rails project[1] and built the communication and built communication and development networks to calculate betweenness centrality, closeness centrality, and degree centrality. In addition, we also considered the number of developers as part of our set of social metrics. Our set of process metrics comprised code churn and number of past file changes. We performed a hierarchical analysis using a multiple linear regression (MLR) model to build the prediction models using the process and social metrics. We also used the relative importance to compare them.

The rest of the paper is organized as follows. In Section 2, we present the study design. Section 3 describes the results obtained from the hierarchical analysis for each network. In Section 4, we compare the relative importance of each metric to predict file change proneness. In Section 5, we present related work. In Section 6, we discuss threats to validity. Conclusions and future work are presented in Section 7.

2 Study Design

The goal of this study is to explore different kinds of social networks and compare social network metrics with process metrics to understand file change proneness. In this sense, we conducted an exploratory study on the Rails project. One of the reasons for choosing Rails is its influential nature [TBLJ13].

We implemented a tool[2] to gather the data and build the communication and development networks. We performed a multicollinearity analysis comparing the metrics pairwise using the data of our dataset. Then, we followed the hierarchical approach used by Bettenburg and Hassan [BH13] to evaluate each prediction model. We studied each beta value, verifying whether the relation had a positive or negative effect, in order to understand the evolution of file change.

[1] Data available on http://github.com/rails/rails
[2] The tool is available at https://github.com/douglasjunior/DouglasJuniorTCC

Finally, we used the relaimpo package[3] to report the relative importance of each metric. In the following subsections, we present the study design in greater detail.

2.1 Data Collection

We collected data from the Ruby on Rails project, hosted by GitHub. Ruby on Rails[4] is a full-stack Model-View-Controller (MVC) framework for database-backed web applications. The Rails project had, in November 2013, 40.203 commits made by 2.498 contributors and representing 169.688 lines of code.

We used the GithubAPI[5] to gather the data. We extracted the history of the source code versioning system and issue tracker from the project. We filtered the data to include only source code-related artifacts, excluding files such as images and documents. We split the data into three intervals of 6 months, from July 2011 until December 2012. We chose these three intervals because they represent the most active periods in terms of contributions. We used a timeframe of 6 months following the approach of previous work conducted by Hong et al. [HKCB11] and [BNG+].

2.2 Process Metrics and Social Networks Analysis

To compare the relative importance of predictors, we collected two process metrics commonly used in the literature (number of past changes and code churn), the number of developers, and three social network metrics computed from communication and development networks (betweenness, closeness, and degree). Process metrics have been used as predictor variables in several studies in the literature [SHAJ12, DLR12] and have been shown to present better results than source code metrics [RD13].

We calculated the number of changes and code churn using commit data from the Git version control system. We also computed the number of developers. To build the development network, we used the number of distinct developers that had made at least one commit on the file. For the communication network, we considered the number of developers that commented on an issue.

We calculated the number of developers, number of past changes, and code churn for every file F that changed during semester S. For each file F, we counted the number of changes in the subsequent interval (S+1). The number of changes was used as the dependent variable. This method is based on the studies conducted by Bird et al.[BNG+] and Bicer et al.[BBC11].

We considered the comments of the issue tracker to build the communication network, and the commits gathered from the source code history to build the development network. For the first network, we assumed a relationship between two developers if both commented an issue in which a given file was changed. To create the second network, we assumed that developers who committed the same file during the period of analysis are linked. The relationships among developers were represented as an undirected graph, in which an unweighted edge was used to represent a link between two developers. Using both communication and development networks,

[3] http://cran.r-project.org/web/packages/relaimpo/
[4] Data available on http://www.ohloh.net/p/rails
[5] Available on developer.github.com/v3/

we computed three different Social Network Analysis (SNA) metrics: degree, closeness, and betweenness centrality [HR05]. JUNG API [6] was used to calculate these metrics.

Betweenness centrality represents the number of shortest paths from all nodes to all others that pass through a given node. Betweenness centrality is useful for explaining the importance of a node in a network [HR05]. Wolf et al. [WSDN09] suggest that people with high betweenness are considered to have more interpersonal influence in the network.

The closeness centrality of a node is the number of steps required to access every other node from a given node. A higher value for a node means it is easier for the node to spread information through the network. Lower values indicate that the node is farther away from all others [HR05]. Degree is the number of edges connected to a node. In this case, a developer's degree is equal to the number of other developers with whom he/she worked or communicated.

The SNA metrics were computed for each developer. To apply these metrics to perform the hierarchical analysis, we used file-based metrics, or metrics calculated on a per-file basis. We adapted the approach proposed by Meneely et al. [MWSO08] to convert a developer-based network metric into a file-based metric. Each file-based metric should reflect the network metrics of developers who changed (committed) the file throughout the semester, or who commented an issue in which the file was changed. To calculate file network metrics, we listed all distinct developers who changed or commented on the file, and calculated the maximum and average value for each developer metric over the semester. For example, the maximum of closeness of file F is the maximum of all developer closeness values for the developers who changed the file F. Values are calculated per-developer, not per-change, so if a developer changed a file twice, his/her metrics would only be used once.

Our set of metrics comprised nine metrics. We classified code churn and number of past changes as process metrics, like [MPS08, DLR12]. In some previous works [SHAJ12, DLR12], the number of developers was considered an organizational metric, or part of the process metric set. We considered the number of developers as a social metric in our dataset.

Instead of adopting number of defects, faults, or bugs as the dependent variable, as some studies have done, we decided to use the number of changes as our dependent variable. We chose this method because bug-tracking systems can also be used to report new features [HJZ13].

In the following subsection, we present details of how we conducted the hierarchical analysis and how we evaluated our models.

2.3 Hierarchical Analysis

The hierarchical analysis applied here was based on statistical models used to investigate the relationship among metrics and to explain the number of future number of changes. Multiple linear regression (MLR) analysis was used to investigate the relationship among the metrics (independent variables) to predict the number of future changes (dependent variables). Regression analysis leads to understand how the value of the dependent variable changes when any independent variable also changes.

In our work, we used the following way to model the number of future changes:

$$numberFutureChanges(filef) = \beta_0 + \beta_{1.CodeChurn} + \beta_{2.NumberOfDevelopers} + \beta_{3.NumberPastChanges} +$$

[6] Documentation available on jung.sourceforge.net/doc/

$\beta_{4.BetweenessAVE} + \beta_{5.ClosenessAVE} + \beta_{6.DegreeAVE} + \beta_{4.BetweenessMAX} + \beta_{5.ClosenessMAX} + \beta_{6.DegreeMAX} + e_i$

The intercept of the model is represented by β_0. The intercept corresponds to the mean value of numberFutureChanges when all of the independent variables are equal to 0. All the values of β_x, with "x" ranging from 1 to 6, are called the regression coefficients. A regression coefficient represents how much the dependent variable is expected to increase (if the coefficient is positive) or decrease (if the coefficient is negative). The last part of the formula (1) e_i is the residual error between the value observed to futureChanges of file F collected during semester s+1 and the value predicted to file F using the MLR formula.

We followed a hierarchical modelling approach to create the MLR models. We started with a baseline model using code churn, a classical defect predictor, and process metrics [BH13]. We then built subsequent models in which we, step-by-step, added our metrics. The metrics that were removed fewer times during the multicollinearity analysis were included first in the model. For each model built, we report the adjusted coefficient of determination R^2. The adjusted R^2 indicates how well the data points fit a line or curve, penalizing the value of R^2 as extra variables are included in the model. An adjusted R^2 of 1 indicates that the regression line perfectly fits the data.

Due to a relatively high amount of skew and kurtosis found in our dataset, we applied a log transformation to each metric, following the same approach applied by [BH13, RD13]. We computed the variance of inflation (VIF) of each metric. In multiple regression, VIF is used as an indicator of multicollinearity. VIF is defined as the reciprocal of tolerance $\frac{1}{1-R^2}$. The value of 10 has been recommended as the maximum level of VIF and this value was adopted in this paper [JBBA09]. To conduct the comparison among metrics, we used relaimpo, an R package that provides several metrics for assessing relative importance in linear models. The metric is lmg, which provides a decomposition of the model-explained variance to compare the contribution of each predictor. In the following section, we present the results of this analysis.

3 Results

We collected and analyzed data from 3 semesters of the Rails project. We built 15 models to analyze development networks and 13 models to analyze communication networks. For each network, we found different results when we performed multicollinearity analysis using VIF.

3.1 Communication Network

To analyze how the communication network can contribute to explain file change proneness, we started our analysis by computing the VIF analysis. To measure the multicollinearity, we needed a complete model, using all metrics together. We removed from the model the metric that presented the highest value for VIF. We chose VIF instead of correlation among pairs of predictors because the pairwise correlations could be small, yet a linear dependence could exist among three or even more variables.

Table 1 presents the values of VIF to the communication network. We begin by showing the complete model (M1) for each semester. We removed from the model the metric that presented

the highest VIF. We conducted this procedure when the metrics presented values higher than 10. Values highlighted in bold show the highest VIF value for each interaction.

Table 1: VIF Analysis of Communication Network in Rails Project

VIF Analysis: Communication Network														
	2011.2				2012.1					2012.2				
log(Yi)	**M1**	**M2**	**M3**	**M4**	**M1**	**M2**	**M3**	**M4**	**M5**	**M1**	**M2**	**M3**	**M4**	**M5**
Churn	3.4	3.4	3.3	3.3	5.32	5.32	5.22	5.21	2.44	2.54	2.53	2.52	2.50	2.49
Developers	24.9	**21.9**	-	-	23.35	21.30	21.27	9.31	7.86	20.18	18.94	8.21	3.02	2.75
PastChanges	7.1	7.0	5.9	5.8	10.95	10.82	10.67	**10.67**	-	3.12	3.10	3.05	3.04	3.03
btwAve	89.4	14.5	10.2	2.9	644.89	86.37	18.53	5.65	5.65	161.60	**143.80**	-	-	-
dgrAve	56.6	15.0	**14.9**	-	466.79	115.14	**73.03**	-	-	115.07	104.44	**28.58**	-	-
clsAve	4.1	4.1	3.4	1.6	19.94	12.54	11.70	2.76	2.67	15.07	14.81	14.10	3.84	3.28
btwMax	**150.0**	-	-	-	**914.22**	-	-	-	-	**382.53**	-	-	-	-
dgrMax	80.3	11.2	3.77	3.6	647.67	**145.5**	-	-	-	167.56	20.74	20.73	**19.70**	-
clsMax	3.0	3.0	2.3	2.3	105.81	102.59	19.34	10.24	9.95	73.39	36.54	17.51	17.17	2.96

We observed that four models were necessary to remove the metrics with VIF higher than 10 for the period of 2011.2. Three metrics were removed from our model in the following sequence: maximum value of betweenness (btwMax), number of developers (developers), and average of degree (dgrAVE). In the following semester (2012.1), we removed four metrics from the model. We removed the maximum betweenness (btwMax), maximum degree (dgrMax), average of degree (dgrAVE), and number of past changes (PastChanges). The only difference between the first and second semester was that we removed btwAve instead of PastChanges. For each semester - considering communication - we did not find a VIF larger than 7.86.

After conducting the multicollinearity analysis, we started our hierarchical analysis with a baseline model (MB). We added the number of distinct developers (NDEVS) and number of past changes (PastChanges). We then added the three average SNA metrics (btwAve, clsAve, dgrAve), and, finally, we included the maximum SNA metrics (btwMax, clsMax, dgrMax). For every stepwise model, we report the regression coefficient for each metric, the $\frac{1}{1-R^2}$ adjusted, the sum of square, and the delta of the sum of square in Table 2.

We conducted the comparison using the ANOVA test. ANOVA returns the goodness-of-fit to compare two different models. The goodness-of-fit was quantified by the sum of squares. If the most complicated model fits worse (highest sum-of-squares) than the simplest model, then we should clearly reject the most complicated model and conclude that the simplest equation fits better.

We report the values of the hierarchical analysis in Table 2. To report the statistical significance, we used this scale: $p < 0.001$, *** $p < 0.01$, ** $p < 0.05$, *. For reasons of space, we report only the hierarchical analysis of 2011.2.

The MB model was chosen to be our baseline model because codechurn was used as a baseline model in other studies, like [BH13]. We also chose codechurn because, considering all the models we built, this metric was not removed from VIF analysis any time.

In model M1, we added the number of past changes. The results of the M1 model shows that including PastChanges improved the R^2 for file changes by 11.98%. The model M1 has a Delta-Sum smaller than the MB ($p < 0.001$). The M2 model included the SNA metrics considering the average. The difference between M2 and M1 is not high, but looking to DeltaSum we can consider that M2 is a valid model and statistically signicant ($p < 0.001$). However, there was not a great improvement in the values of R^2 (increasing of 1.96%). The last model, M3, is statistically significant ($p < 0.001$) and R^2 value increased 2.4%. The best value to explain file changes was

Table 2: Hierarchical Analysis of Communication Network 2011.2

Communication Network - MLR - Second Semester 2011								
	MB		**M1**		**M2**		**M3**	
CHURN	0.28	***	0.06	***	0.05	***	0.05	***
NDEVS	removed Multicollinearity Analysis (second round)							
PastChanges			0.67	***	0.61	***	0.36	***
dgrAve	removed Multicollinearity Analysis (third round)							
btwAve					-0.13	***	-0.35	**
clsAve					-5.49	***	-6.49	***
dgrMax							0.45	
btwMax	removed Multicollinearity Analysis (first round)							
clsMax							1.59	***
R^2 adjusted	47.17%	***	59.15%	***	61.11%	***	63.51%	***
Sum of Sq	1135.2		877.43		846.09		784.84	
DeltaSum			257.77	***	31.34	***	61.25	***

obtained using M3 (63.51%), meanwhile the most relevant improvement was obtained from MB to M1 (11.98%).

For the following two semesters of 2012, we found a similar explanation to predict file changes. For the first semester, we built four models. The MB model achieved 56.90% of R^2. The M1, M2, and M3 models had R^2 values of 63.85%, 67.46%, 69.31%, respectively. For the second semester of 2012, we built five models. The VIF analysis did not remove the number of developers or past changes as observed in the previous two semesters. The MB model achieved 31.78% of R^2. Models M1, M2, M3, and M4 had R^2 values of 49.70%, 67.02%, 67.84%, and 68.81%, respectively. All the metrics used to build models for 2012.1 and 2012.2 were statistically significant ($p < 0.001$).

To explain the relation of each metric, we conducted a signal analysis. Table 3 presents the results. The positive sign (+) indicates that the metric is positively related to the amount of file changes, while the negative sign (-) indicates a negative relation. The number zero (0) indicates a neutral relation. Bold values and the gray color indicate that the values did not pass the signicance test ($p < 0.001$).

Table 3: Signal Analysis

Signal Analysis													
	2011/2				**2012/1**				**2012/2**				
	MB	**M1**	**M2**	**M3**	**MB**	**M1**	**M2**	**M3**	**MB**	**M1**	**M2**	**M3**	**M4**
CHURN	+	+	+	+	+	+	+	+	+	+	**0**	**0**	**0**
NDEVS						+	+	+		+	+	+	+
PastChanges		+	+	+							+	+	+
dgrAve													
btwAve			-	-			+	-					
clsAve			-	-			+	**-**				+	+
dgrMax				**+**									
btwMax													
clsMax				+				+					-

We observed that code churn changed the positive effect to neutral when we added PastChanges and number of developers in M2, M3, and M4 for the second semester of 2012. ClsAve and btwAve used in the M2 and M3 models of 2012.1 presented a negative impact on the model when we added the clsMax. The same behavior was not observed in M3 for 2011.2.

Finally, we computed the relative importance of each metric. Table 4 presents the values for the last two models for each semester. We chose the last two models because they had more variables together and we wanted to investigate how each metric contributes to the model. We used the letters (A-F) to rank the position of relative importance. The letter A indicates better

performance compared to the other metrics. The value VIF indicates that it was removed during the multicollinearity analysis.

Table 4: Relative Importance - Communication Network

Relative Importance - Communication Network						
	2011.2		2012.1		2012.2	
	Model 2	Model 3	Model 2	Model 3	Model 3	Model 4
CHURN	B	B	A	A	C	C
NDEVS	*VIF*	*VIF*	A	B	B	B
PastChanges	A	A	*VIF*	*VIF*	A	A
dgrAve	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*
btwAve	C	C	C	D	*VIF*	*VIF*
clsAve	C	F	D	E	D	E
dgrMax	*VIF*	C	*VIF*	*VIF*	*VIF*	*VIF*
btwMax	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*
clsMax	*VIF*	C	*VIF*	B	*VIF*	D

Considering the communication network, the SNA metrics did not perform well when compared to code churn, past changes, and number of developers. None of the SNA metric were better than one of those three metrics. The dgrAVE and btwMAX were always removed by VIF analysis. In the following subsection, we present the results of the development network.

3.2 Development network

We conducted the same experiments to compare the metrics using the development network. Table 5 presents the values of VIF for each semester analyzed.

Table 5: VIF Analysis for Development Network from Rails Project

VIF Analysis: development network											
	2011.2				2012.1				2012.1		
log(Yi)	Model 1	Model 2	Model 3	Model 4	Model 1	Model 2	Model 3	Model 4	Model 1	Model 2	Model 3
Churn	2.01	2.01	2.01	2.00	2.24	2.20	2.17	2.08	2.21	2.21	2.19
Developers	12.93	12.91	11.64	9.00	12.85	12.53	8.96	8.77	8.06	4.06	4.04
PastChanges	9.83	9.83	9.61	9.21	6.94	6.94	6.93	6.83	3.84	3.84	3.74
btwAve	65.94	31.22	**22.54**	-	112.42	**110.33**	-	-	**49.94**	-	-
dgrAve	42.70	15.05	11.78	2.09	34.09	33.59	**33.58**	2.80	34.93	**33.87**	-
clsAve	**74.53**	-	-	-	59.81	59.81	32.23	-	41.04	31.44	1.87
btwMax	45.21	**39.45**	-	-	**919.97**	-	-	-	33.04	6.91	6.89
dgrMax	48.82	25.90	3.53	1.71	658.55	69.13	12.63	4.45	12.06	11.88	4.07
clsMax	5.53	2.34	1.79	1.29	42.45	9.24	8.77	5.76	12.01	11.62	8.29

We built four models to remove metrics with VIF greater than 10. Three metrics were removed: average of closeness (dgrAVE), maximum betweeneness (btwMax), and average of betweeneness (btwAve). For the following semester (2012.1), we removed btwMax, clsAve, and btwAve. For 2012.2, we removed only two metrics: btwAve, and dgrAve. The highest VIF value was 9.21.

The results of the hierarchical analysis are presented in Table 6. Due to space restriction we only report the values for the hierarchical analysis from the 2011.2 development network.

Model MB is a baseline model. M1 includes the number of past changes. The results of the M1 model show that including number of developers improved the explanation (R^2) of file changes of the model in 14.04%. The M1 model has small value for DeltaSum and the model was statistically significant ($p < 0.001$). The M2 model added past changes. The difference between M2 and M1 was 5.91%. When we included the past changes, the number of developers ceased to be significant for the model. The model did not improve its performance when we

Table 6: Hierarchical analysis for Development Network 2011.2

Hierarchical analysis from Communication Network 2011.2										
	MB		**M1**		**M2**		**M3**		**M4**	
CHURN	0.36	***	0.18	***	0.11	***	0.11	***	0.10	***
NDEVS			0.93	***	-0.13		-0.09		0.05	
PastChanges					1.03	***	1.02	***	0.98	***
dgrAve							0.12	.	0.42	***
btwAve	removed Multicollinearity Analysis (second round)									
clsAve	removed Multicollinearity Analysis (third round)									
dgrMax									-0.38	*
btwMax	removed Multicollinearity Analysis (first round)									
clsMax									-6.16	***
R2 adjusted	39.90%	***	53.94%	***	59.15%	***	59.31%	***	60.22%	***
Sum of Sq	745.36		570.88		506.01		504.99		493.68	
DeltaSum			174.48	***	64.87	***	1.02	***	11.31	***

included the dgrAve. The same happened when we included clsMax and dgrMax.

For the first semester of 2012 (2012.1), we built 5 models. MB achieved 35.36% for R^2. M1, M2, M3, and M4 had R^2 values of 52.06%, 52.36%, 54.86% and 54.93%, respectively. For the second semester of 2012 (2012.2), we also built five models. MB achieved 32.23% to R^2. M1, M2, M3, and M4 R^2 values of 54.20%, 62.50%, 63.05% and 63.44%, respectively. All the models built for 2012.1 and 2012.2 had statistical significance ($p < 0.001$).

During the signal analysis, we observed that code churn changed the signal for M2, M3, and M5 during the 2012.2 period. A similar pattern was observed in the communication network: code churn changeed its effect from positive to neutral when we added past changes and number of developers.

SNA metrics computed for development networks presented negative effects when predicting change proneness. However, the beta coefficient values for them often presented no statistical significance. This behavior indicates that these metrics do not increase the values of R^2 when they are added to a model.

Table 7 presents the relative importance of each predictor using the development network.

Table 7: Relative Importance for Development Network

Relative Importance - development network						
	2011.2		**2012.1**		**2012.2**	
	Model 3	**Model 4**	**Model 3**	**Model 4**	**Model 2**	**Model 3**
CHURN	B	C	C	C	C	C
NDEVS	B	B	A	A	B	B
PastChanges	A	A	B	B	A	A
dgrAve	D	D	*VIF*	*VIF*	*VIF*	*VIF*
btwAve	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*
clsAve	*VIF*	*VIF*	D	D	D	D
dgrMax	*VIF*	F	*VIF*	F	*VIF*	G
btwMax	*VIF*	*VIF*	*VIF*	*VIF*	*VIF*	D
clsMax	*VIF*	D	*VIF*	D	*VIF*	D

No metric calculated using the networks performed better than code churn, number of past changes, and number of developers. DgrAVE and btwMAX were always removed by VIF analysis. The same pattern was observed for the communication networks.

4 Comparing Communication and Development Networks

In this paper, we investigated two research questions: RQ1: What are the differences when process and social metrics related to communication and cooperation networks are used to predict

file change proneness? and RQ2: What is the relative importance of each metric to predict file change proneness?

4.1 RQ1: What are the differences when process and social metrics related to communication and cooperation networks are used to predict file change proneness?

We performed analyses comparing the relation among process and social metrics considering the evolution of changes for each network. We presented in Subsections 3.1 and 3.2 more details about the relation of each metric set for each kind of network. We provided a multicollinearity analysis using VIF, the results of each hierarchical analysis conducted during a semester, and the signal analysis.

Table 8 presents the results of the hierarchical analysis using MLR models to summarize RQ1. The table presents the worst result and the best result for each network. The results compare the baseline model with the last model created. The last model contains all the variables that have VIF smaller than 10. The diff column shows the difference between worst and best models in each semester.

Table 8: Summarizing the Hierarchical Analysis of Communication and Development Networks: Worst and Best Results

Summarizing the adjusted coefficient of determination of Hierarchical analysis				
		worst result	better result	diff
Communication	2011.2	47.1	63.5	16.4
	2012.1	56.9	69.3	12.4
	2012.2	31.7	68.8	37.1
Development	2011.2	39.9	60.2	20.3
	2012.1	35.3	54.9	19.6
	2012.2	32.3	63.4	31.1

Considering the communication networks, the difference between the baseline model (worst result) and the last model built (best result) was smaller to the values obtained from development networks.

The hierarchical analysis using communication networks showed that past changes and number of developers improved the R^2 from 6.95% to 17.92% to create a new model. When we added the SNA metrics, the improvement to R^2 was smaller, from 0.86% to 3.61%. Considering the development network, past changes and number of developers, R^2 increased from 5.21% to 21.97%. The SNA metrics increased R^2 from 0.07% to 2.50%.

Process metrics (code churn and number of past changes) and the social metric (number of developer) **performed better** than SNA metrics to predict the evolution of file changes. The hierarchical analysis showed that the number of changes and number of developer always increased the values of R^2. The contribution of SNA metrics to the models was smaller.

4.2 RQ2: What is the relative importance of each metric to predict file change proneness?

To assess the impact of each predictor individually, we used the package relaimpo. We ran the analysis using the metric lmg, congured to generate the bootstrap of a 95% confidence interval for R^2 in the MLR and to present a ranked comparison of each predictor and its relative importance.

Table 9: Ranking Comparison between Communication and Development Network

Ranking Comparison												
	Communication						Contribution					
	A	B	C	D	E	VIF	A	B	C	D	E	VIF
CHURN	2	2	2	0	0	0	0	1	5	0	0	0
NDEVS	1	3	0	0	0	2	2	4	0	0	0	0
PastChanges	4	0	0	0	0	2	4	2	0	0	0	0
dgrAve	0	0	0	0	0	6	0	0	0	2	0	4
btwAve	0	0	3	1	0	2	0	0	0	0	0	6
clsAve	0	0	1	2	2	0	0	0	0	4	0	2
dgrMax	0	0	1	0	0	5	0	0	0	0	0	3
btwMax	0	0	0	0	0	6	0	0	0	1	0	5
clsMax	0	1	1	1	0	3	0	0	0	3	0	3

Code churn was not excluded from the VIF analysis. This led us to choose this metric as the base model. Number of developers and number of past changes were removed twice for the communication network. However, they were not removed from the development network.

Considering the SNA metrics, using the average aggregation, degree average (dgrAVE) was always excluded in VIF analysis. Degree value was always very close to the number of developers, which probably excluded this metric during multicollinearity analysis. The metric btwAVE was excluded more times than clsAVe, however btwAve was more relevant when it was used in the models.

Looking at the relative importance, betweeness average explained 10.94% of evolution of changes for 2011.1 using the communication network. It was able to explain 4.44% for 2012.1. Closeness average explained 9.56% and 1.63% respectively for 2012.1 and 2012.2.

The maximum aggregation presented better result than average metrics. However, they were excluded more often during VIF analysis. Considering communication networks, drgMAX and clsMAX could explain 12.76% and 11.34% of evolution of changes for 2011.1. clsMax could explain 30.01% for 2012.1 and 5.28% for 2012.2. SNA metrics had lower performance using the development network. All metrics presented at least two exclusions (VIF analysis). None of them could explain more than 6.78% of the changes in files.

The relative importance of code churn ranged between 18.1% and 45.4% in both networks. Number of developer achieved 22.92% to 46.76% using communication networks, and 30.06% to 38.64% using development networks. Finally, the number of past changes contributed to explain 32.66% to 53.41% in communication networks and 29.39% to 45.52% in development networks.

To provide a better understanding of each metric, we conducted another multivariate analysis. This enabled us to explain the specific contribution of each metric. Factor analysis is a statistical method used to describe variability among variables (metrics). We can use factor analysis when it is necessary to know about underlying factors. When variables are numerical, it is possible perform a factor analysis using PCA (Principal Components Analysis) [HLP10].

Figure 1 depicts the PCA analysis of the communication network for 2011.1. On the left side, we present the samples plotted. They represent the TOP-60 files that changed during the period. On the right side, we present the metrics. Each vector represents one metric. If the angle between two vectors is small, this indicates a high correlation between the metrics.

Figure 1 shows that the average closeness (clsAve) is important to explain a group of files that no other metric could explain. For example, sample (file) 57 presented the highest closeness to the dataset. On the other hand, only two developers touched the file. Files close to sample 57

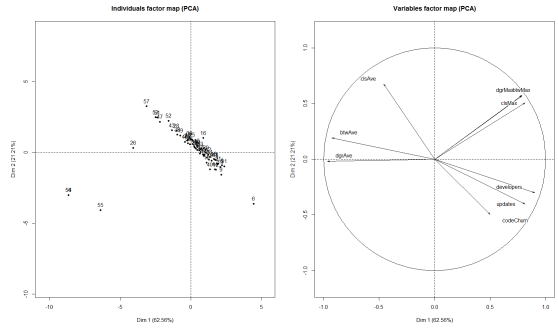

Figure 1: PCA analysis to communication network 2011.1

presented the same behavior. This means that there is a group of files affected by the number of developers and another group of files affected by the value of clsAve. We can see that the files that present a greater number of developers tend to have lower value of clsAve and vice-versa. Although, most of the variability of the files was explained by the number of developers, clsAve could explain another particular group of files. It is important to determine the contribution of clsAve to explain the file changes.

We also determined that maximum degree, closeness, and betweeness are highly correlated to each other, which explains why they were frequently removed from the hierarchical analysis. The average of betweenness and degree were less correlated, but they explained no group of files. We also observed some outliers, for example, sample 55.

We performed the same factor analysis to observe each semester, and the development networks. We found that metrics bear influence in different ways, because the vectors (metrics) point in various directions, and we did not observe a single pattern.

5 Related Work

The modication of a file by several developers may make that file prone to change and harder to modify. A change that touches a file that has been modified by many different developers can have more risk [SHAJ12].

Bettenburg and Hassan[BH13] found that statistical models based on social information have a similar degree of explanatory power as traditional models (source code metrics and process metrics). They studied Mozilla and Eclipse projects and collected data from issue tracking and version control repositories. Some studies use communication and development networks. They compute social network metrics and use these metrics as input to build prediction models. Meneely et al. [MWSO08] examined the development network derived from code churn information. They conducted a case study, and found that there is a significant correlation between file-based development network metrics and failures. Wolf et al. [WSDN09] reported results indicating that developer communication plays an important role in software quality. They predict build failure on IBM's Jazz project yielding recall values between 55% and 75%, and precision values between 50% to 76%. Bird et al. [BNG+] evidence the influence of combined sociotech-

nical software networks on the fault-proneness of individual software components. They reported results using precision and recall around of 85%. Bicer et al. [BBC11] created models to predict defects on IBM's Jazz project and Drupal. Their results revealed that compared to other metrics such as churn metrics, social network metrics either considerably decreases high false alarm rates.

The studies presented in this section use a different set of metrics, different ways to build the networks, and used different projects. They showed that social metrics can be good predictors, but it is hard to know in which conditions it is possible to use these metrics. Many of these projects are commercial, where one expects a better organization of the software process used.

6 Threats of Validity

In this section, we discuss various factors that may have influenced this work.

Set of metrics: our set of metrics is small compared to other papers. We considered just two process metrics, the number of developers that commented on an issue or committed a file, and three different SNA metrics. However, we included in our process set the churn metric that is usually used as a baseline model. This work is our first effort to compare process metrics and SNA metrics. As the results are encouraging, some other metrics can be included to explain the evolution of file changes, as well to test our findings in a large-scale setting.

Building Network: We built social networks using undirected edges and without weight. Some SNA metrics may have performed better if we used weighted edges and directed graphs. We intend to investigate this in future work.

Intervals selection: We used 6 months as the interval with which to compute the metrics. We know that different intervals of analysis can lead to different results. We checked the correlation between commits and number of developers and we found that there is strong correlation between these two measures for 2011.2 (0.72) and 2012.1 (0.86). For the last semester, a weak correlation (0.26) was identified. Since we are interested in studying the evolution of changes, instead of defects, we realize that file can undergo a considerable number of changes in a six-month interval. In the Rails project, we observed that our dataset doubled the amount of change from one semester to another, including the TOP-100 files that were most changed. We ran a correlation between number of commits and number of developers. We found a high correlation for 2011.2 and 2012.2, but we found a weak correlation for 2012.2.

Generalizability: This study aimed to explore the use of metrics and statistical techniques for predicting file change proneness. Additional studies are necessary to generalize the results.

7 Conclusions and further work

In this paper, we explored statistical relations to compare process and social metrics considering file change proneness. We used a set of metrics to compare the results using two different networks: communication and development.

We found that code churn, number of past changes, and number of developers could explain the change proneness for the Rails project better than SNA metrics. Considering the SNA metrics, the best results were obtained using communication networks. Using the average of betwee-

ness (btwAve) we could explain 10.64% of change proneness. Using development networks, we found that average of closeness (clsAve) was the best SNA metric, explaining 6.7%. However, when we added SNA metrics during the hierarchical analysis, the value of R^2 did not increase significantly. We also conducted a factor analysis using PCA to show that some files can be influenced by social metrics, even if they have not added great improvement to build new models during the hierarchical analysis. In the future work, we plan to extend the analysis to other projects and investigate in-depth the influence of social metrics to explain software evolution - for example, to explain the social of predicting pull requests for specific group of files.

Acknowledgements: The authors of this work thank Fundacao Araucaria and NAWEB for the financial support. Marco Aurelio Gerosa receives individual grant from the Brazilian National Research Council (CNPq) and FAPESP. Igor Wiese and Igor Steinmacher receive grants from CAPES (Process BEX 2039-13-3 and BEX 2038-13-7).

Bibliography

[BBC11] S. Bicer, A. B. Bener, B. Cauglayan. Defect prediction using social network analysis on issue repositories. In *Proceedings of the 2011 International Conference on Software and Systems Process*. ICSSP '11. 2011.

[BH13] N. Bettenburg, A. E. Hassan. Studying the impact of social interactions on software quality. *Empirical Software Engineering* 18(2):375–431, 2013.

[BNG+] C. Bird, N. Nagappan, H. Gall, B. Murphy, P. Devanbu. Putting It All Together: Using Socio-technical Networks to Predict Failures. In *Proceedings of the 2009 20th International Symposium on Software Reliability Engineering*.

[DLR12] M. D'Ambros, M. Lanza, R. Robbes. Evaluating defect prediction approaches: a benchmark and an extensive comparison. *Empirical Software Engineering* 17(4-5):531–577, 2012.

[GBL+13] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, A. v. Deursen. Communication in open source software development mailing lists. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. MSR '13. 2013.

[HBB+12] T. Hall, S. Beecham, D. Bowes, D. Gray, S. Counsell. A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *Software Engineering, IEEE Transactions on* 38(6):1276–1304, 2012.

[HJZ13] K. Herzig, S. Just, A. Zeller. Its not a bug, its a feature: how misclassification impacts bug prediction. In *Proceedings of the 2013 International Conference on Software Engineering*. ICSE '13. 2013.

[HKCB11] Q. Hong, S. Kim, S. C. Cheung, C. Bird. Understanding a developer social network and its evolution. In *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*. ICSM '11, pp. 323–332. 2011.

[HLP10] F. Husson, S. Lê, J. Pagès. *Exploratory Multivariate Analysis by Example Using R*. Chapman & Hall/CRC, 2010.

[HR05] R. A. Hanneman, M. Riddle. *Introduction to Social Network Methods*. 2005.

[JBBA09] J. F. H. Jr, W. C. Black, B. J. Babin, R. E. Anderson. *Multivariate Data Analysis*. Volume 7 edition. 2009.

[LB85] M. M. Lehman, L. A. Belady (eds.). *Program evolution: processes of software change*. 1985.

[MPS08] R. Moser, W. Pedrycz, G. Succi. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In *Proceedings of the 30th international conference on Software engineering*. ICSE '08. 2008.

[MWSO08] A. Meneely, L. Williams, W. Snipes, J. A. Osborne. Predicting failures with developer networks and social network analysis. In *SIGSOFT FSE*. Pp. 13–23. 2008.

[NMB08] N. Nagappan, B. Murphy, V. R. Basili. The influence of organizational structure on software quality: an empirical case study. In *International Conference on Software Engineering (ICSE)*. Pp. 521–530. 2008.

[RD13] F. Rahman, P. T. Devanbu. How, and why, process metrics are better. In *ICSE*. Pp. 432–441. 2013.

[Sch00] N. F. Schneidewind. Software quality control and prediction model for maintenance. Volume 9(1-2). 2000.

[SHAJ12] E. Shihab, A. E. Hassan, B. Adams, Z. M. Jiang. An industrial study on the risk of software changes. In *SIGSOFT FSE*. P. 62. 2012.

[TBLJ13] F. Thung, T. F. Bissyande, D. Lo, L. Jiang. Network Structure of Social Coding in GitHub. *2011 15th European Conference on Software Maintenance and Reengineering*, pp. 323–326, 2013.

[WOB07] E. J. Weyuker, T. J. Ostrand, R. M. Bell. Using Developer Information As a Factor for Fault Prediction. In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*. PROMISE '07. IEEE Computer Society, Washington, DC, USA, 2007.

[WSDN09] T. Wolf, A. Schroter, D. Damian, T. Nguyen. Predicting build failures using social network analysis on developer communication. In *Proceedings of the 31st International Conference on Software Engineering*. ICSE '09. 2009.