Organizational Volatility and Developer Productivity

Audris Mockus
Avaya Labs Research
233 Mt Airy Rd, Basking Ridge, NJ
audris@avaya.com

Abstract

The key premise of an organization is to allow more efficient production, including production of software. To achieve that, an organization defines roles and reporting relationships. Therefore, changes in organization's structure are likely to affect organization's productivity. We investigate several measures of organization and organizational change and relate them to developer productivity. We find that the proximity to an organizational change is strongly associated with a reduced developer productivity.

1. Introduction

The key premise of Organization is that it allows more efficient production [22], including production of software. Organizational design defines roles, processes, and formal reporting relationships to make organization more productive. Therefore, changes in organizational design are likely to affect organization's productivity, including the productivity of development processes. Furthermore, according to Conway's law [10], the structure of an organization is reflected in the structure of the systems it designs. Therefore, the change in a software organizations may have particularly adverse effects if the changed organization no longer matches the initial design of the software. In particular, it was shown that tasks requiring participation from multiple sites take almost three times longer to complete [13]. Our investigation is focused on studying how the organizational change, rather than organizational structure, affect software development. Past work investigating software organizations found that effective organizations have very low worker turnover [9] due to what authors call "social capital" invested in the workforce through culture of trust and respect, generous benefits, and recognition of importance of peoples personal lives. It is, therefore, of interest to quantify the benefits of low turnover or costs of high volatility.

Because organizational churn relates to people entering and leaving the organization we used organizational socialization theory [21] to frame our hypotheses on how organizational volatility may affect the productivity of software developers. As in [15], we assume that productivity reflects innovation. We have borrowed and extended measures of organizational change based on archival records as considered in, for example, [12].

At a high level, the primary positive results of volatility would be the innovation brought by incoming people. The negative side involves the overhead for the existing team to train new developers, the initial lack of experience of new developers, and the gaps in tacit knowledge produced by departures of experienced developers. From the emotional perspective, the positive results would include increase in the usage of direct action (focusing more energy on one's job) coping strategies that would lead to increase in performance. The negative side of the emotional responses would involve increase in withdrawal strategies (avoidance and disengagement) and decreased health of the employees that stay, leading to reduced productivity.

We propose a variety of measures of organizational volatility, obtain them for a multi-year sample involving more than one thousand developers, and test our propositions using a linear model.

We start by discussing related work in Section 2 and present the context of the study and data collection in Section 3. The hypotheses are presented in Section 4, operationalizations in Section 5, and results in Section 6. We conclude in Section 8

2. Related work

Organizational change has significant amount of reserch- and practice-oriented literature. While much of it is action oriented and prescriptive, little is devoted to methods to measure organizational volatility from archival data, despite Webb et al [23] raising the issue as early as 1966 well summarized by a Chinese proverb: "the palest ink is clearer that the best memory." A notable exception is work by Geisler [12] that proposes a number of organizational change measures based on archival records. Unlike the measures we use that are based on human resource systems, that work required manual search through document records.

A particular form of volatility related to downsizing has received substantial attention. Notably, the results appear to be similar to what would be predicted from organizational socialization theory as described in Section 4. In particular, a study of mid-level managers and executives in Canadian civil service [3] found that downsizing has been associated with reduction in productivity of the survivors (individuals that remain), increase of their workload demands, increase in escape coping strategies, and increase in the incidence health problem symptoms. The study used four surveys — one prior to, two during, and fourth after the downsizing over a three year period. The four survey response rates were 58, 46, 44, and 43 percent correspondingly. The analysis distinguished control-based coping strategies that included positive thinking (recasting the downsizing event as a challenge), direct-action (focusing more energy on your job), and instrumental support (seeking information from others), and escape coping strategies that included avoidance of the situation and disengagement (putting less effort into work and spending more time on activities outside work). Other empirical evidence also shows that downsizing is associated with decreased job performance, for example, [2]. In addition to studying developers, unlike in prior work, we also look at more general volatility that includes downsizing. Furthermore, we look at a full sample of developers without an inconvenience of large non-response rates and our observations cover the entire period without restrictions to survey administration times.

A yet another view on volatility touches a theoretical framework developed for emotional labor. Emotional labor was initially developed for airline attendants [14] and was initially defined as "the management of feelings to create publicly observable facial and body display." It was later substantially generalized, because most corporate work requires some level of emotional labor. The aspect salient to volatility relates to the empirical findings in, for example, [8], showing that surface acting (modifying and controlling emotional expressions) requires much more effort than deep acting (consciously managing feelings) and thus reduces job performance. Furthermore, the downsizing, which is a common cause of volatility, produces negative emotions that are much harder to control via deep acting than positive emotions, see, for example, [7].

A macroscopic view of the effects of turnover on organization's performance is presented in, for example, [5]. Authors study turnover in a number of young high-technology companies and find that changes in employment models embraced by organizational leaders increase turnover and adversely affect subsequent organizational performance. They argue that their results indicate that cultural blueprints of organization embodied by employment models are persistent and the organizations would be destabilized and disrupted if these blueprints were to change. We consider phenomena at a much finer micro level of individual performance. However, it is worth noting that the company under study has gone through a radical change in management during the course of being taken into private ownership. Therefore, some of the macroscopic effects observed in [5] may be relevant in our case as well.

3. Context and data sources

We investigate software development in Avaya with many past and present projects of various sizes and types involving more than two thousand developers. Two primary sources of data were utilized in the study. The changes to the source code were obtained from a variety of version control systems used in Avaya, including SCCS, ClearCase, CVS, and SubVersion. The data were cleaned to eliminate administrative changes (changes made for the purpose other than to enhance or fix the product) using a variety of techniques appropriate for each system and each project. For example, the initial delta for branches in SCCS or in CearCase that do not change the source code were excluded. The data cleaning and validation was done to support project measurement and prior studies, for example, [17] and, therefore is not described in more detail here. We used developer ID making the change and the date of change to calculate developer productivity.

The second source of data was an organizational database (POST) that lists individuals, their organization, and contact information. We had collected frequent snapshots of this data over a period of seven years. The primary purpose of this data was to establish the identity of each developer and to calculate measures of organizational volatility. As with any other source of data, it had its share of anomalies and issues. First, developer logins were not always identical to email handles in POST. Second, logins have changed over time for some developers because a recent policy required logins to match email handles. Third, the email handles and even organizational IDs have changed for some developers. For example, offshore developers who started at a US location and later went back to their permanent offshore site got a new ID at their home site. To deal with these issues we used a NIS database (snapshots of which we have also collected over seven years) that mapped login to the organizational ID and the full name of the person authorized to use the login. This extra piece of information allowed us to establish the identities of developers over time despite changes in the organizational ID's, email handles, and sometimes even names (for example, as a result of a marriage). We have used these sources of data to map logins and organizational IDs to unique numeric IDs identifying each participant. These unique IDs were then substituted for logins in the code change data and for organization IDs in the POST data to normalize identity information and to provide more privacy (some developers could be recognized from their login). While there were more than ten thousand emploees in POST, only around two thousand of them modified the source code.

4. Hypotheses

To formulate effects of organizational change on developer productivity we borrow from a variety of theoretical frameworks and empirical results. Based on the extensive work discussed in Section 2, it is reasonable to expect that:

Proposition 1 *Organizational volatility affects productivity of the development process*

The manner in which the volatility affects productivity depends on a variety of factors and theoretical assumptions. In particular, the arrival of new experienced member is likely to increase productivity through innovations they may bring [21]. Therefore,

Proposition 2 *New experienced members would bring innovations and, therefore, have higher productivity or find new ways to improve quality.*

On the other hand, based on the same reasoning, new inexperienced members would require training reducing productivity of existing developers. A variety of training methods are described in, for example, [24].

Proposition 3 *New inexperienced members would require training reducing productivity of existing developers.*

As was shown in, for example, [18], it may take new developers several years to reach full productivity in a legacy system. Furthermore, according to [24] new members are likely to participate in peripheral tasks, further reducing the their productivity.

Proposition 4 *New inexperienced members would be less productive and more likely to introduce defects.*

Finally, the departure of experienced members may leave gaps in the tacit knowledge, see, for example, [19]. Such gaps, may lead the remaining team members to make suboptimal or even disastrous design and implementation decisions.

Proposition 5 *Outgoing members would leave gaps in the tacit knowledge, making suboptimal design and implementation decisions more likely by the remaining team.*

Furthermore, if such departures are related to unfavorable business conditions and downsizing, that may exert additional stress on the remaining employees thus reducing their performance for reasons discussed in Section 2.

Proposition 6 *Layoff related departures may exert emotional toll and increase performance reducing coping strategies on all remaining developers.*

5. Measures of organization and its change

We followed [12] in measuring organizational change. However, our data and our hypotheses were substantially different, therefore the operationalizations of the measures also have little resemblance. One of the factors that we considered was the reporting hierarchy of the organization to capture changes in organizational structure brought by a reorganization. The staff reductions discussed above are often associated with the reorganization. Because our analysis has a response corresponding to the productivity of an individual developer, we also look at the organizational features of comparable size. Therefore, we look at the number of employees in the organizational unit developer belongs to. The organizational unit was operationalized in two ways: through a supervisor ID and through an organizational ID. The analysis results are very similar for both of these operationalizations. In Table 1 we present one of the operationalizations. Alternative operationalization may be obtained by replacing words “supervisor ID” by “organizational ID.”

At the high level, we looked at the proximity to the organizational change because, for example, [3] found that performance reductions were transient and did not manifest themselves either well before or well after the downsizing. We also anticipate that the extent of the reorganization measured in the number of people coming and leaving the organization should have effects described in Propositions 2, 3, 4, 5, and 6. The number of new recruits are needed to investigate Propositions 3 and 4, and the number of employees leaving the organization is needed to test Proposition 6

We may need to adjust for the possible variation in developers’ responsibilities by counting the number of employees reporting to a developer. As was found in, for example, [15], the size of the project affects the productivity of new developers. Therefore we may need to adjust for organization size, product, and geographic location. Due to well-known variations among developers, see, for example, [11], we may need to adjust for the individual developer or their organization.

Our response variable operationalized developer productivity. Conceptually, the productivity of a developer is the number of product units (output) produced over some unit of effort (inputs). For commercial developers who are employed full-time, the inputs may be roughly approximated

Concept	Operationalization
Proximity in time to the organizational change	Time (in years) until the next and after the last change in the organization ID
Size of the reorganization	Number of employees transferred into and out of the organization from (other units in Avaya) over past two months
New recruits	Number of employees hired into the organization over past two months
Departures	Number of employees leaving Avaya from the organization over past two months
Developers level in the reporting hierarchy	Number of reports (subtree)
Size of the organization	Number of employees with the same organization ID
Other factors	Product, Location, Organization ID, Developer ID

Table 1. Concepts and their operationalizations.

by developer time (staff-months) multiplied by salary and other employment costs. However, unlike in manufacturing, in software the product units are typically not well defined. Most commonly used measures of software output lines of code (LOC) or Non-Commentary Lines of Code (NCSL) are easy to obtain but tend to have numerous drawbacks and have to be adjusted for system size, staffing levels, development capability, programming language, the extent of reuse, and the type of development activity (see, e.g., [6, 11]). Another commonly used measure of output is Function Points [1]. However, it is more difficult to calculate and was not used in this organization.

Therefore, we chose to use the number of changes per staff-month as a pragmatic measure of developer productivity, because it was readily available (similar to NCSL) and has been successfully used in the past [4, 16]. In particular, the study in [4] has investigated a relationship between software features that are sold to customers and the number of changes needed to implement that feature and found a strong relationship between changes and sellable functionality. The summary of developer experience with respect to a part of the system expressed in the number of changes was found to reflect developers' and managers' subjective perceptions of expertise [16].

Because we are comparing the productivity of the same developer under the conditions of high and low organizational volatility, the comparisons automatically adjust for the inherent differences in making changes to different applications, using different programming languages, and other code- or individual-related factors.

6. Results

To test propositions in Section 4 we collected data for the period between January 2004 and December 2008 for the entire organization. Because our primary concern is with software developers, we only use organizational measures that are directly related to the developers who made changes to the source code. We considered only developers that have made changes for at least five months, who worked on 58 largest products, and in eleven largest development sites to exclude unusual cases. The data covered 1227 developers who had 334 distinct supervisors, worked on 58 different projects, and were based in 11 geographic sites.

Table 2 shows the analysis of variance and regression results for the model with response being the logarithm of the number of changes a developer made each month. Predictors were the developer ID, their supervisor ID for that month (more than half of the developers had at least two supervisors over the considered period) to adjust for individual and organizational differences. The remaining predictors were used to test our propositions in Section 4 and included the number of new recruits within the last two months, the time (in years) until the next organizational change, the time after the last organizational change, the number of people reporting to the developer, and the number of people transferring to other parts of the company or leaving the company within the last two months.

Table 2 presents ANOVA results indicating the degrees of freedom and sum of squares for each variable. The values for ANOVA F-statistics and the associated p-values are also displayed. Finally, for the numeric variables, the fitted regression coefficients and associated p-values calculated from the regression's t-statistics are also presented[1].

Based on the ANOVA, all predictors except for the number leaving the organization in the past two months were significant (p-values for F-statistics are less than 0.05). This result does not support Proposition 5. Possibly, to see the effect of the knowledge gaps we may look at time periods longer than two months. Alternatively, we may need to inspect how the centrality of the tasks changes after such departures to fully assess the impact of the knowledge loss. Finally, we may need to focus not simply on the number

[1]The regression and ANOVA were done using R [20] package

Predictor	Df	Sum Sq	Mean Sq	F value	Pr(>F)	Coefficient	Pr($> t$)
Developer ID	1226	33918.00	27.67	17.10	0.00		
Supervisor ID	267	1752.13	6.56	4.05	0.00		
$\log(\text{Newcomers})$	1	12.51	12.51	7.73	0.01	-0.02	0.16
$\log(\text{From prior})$	1	17.14	17.14	10.59	0.00	0.10	0.00
$\log(\text{Until next})$	1	109.34	109.34	67.56	0.00	0.13	0.00
$\log(\text{Reports} + 1)$	1	14.23	14.23	8.79	0.00	-0.25	0.00
$\log(\text{Left} + \text{Transfred} + 1)$	1	0.00	0.00	0.00	0.98	-0.00	0.98
Residuals	24004	38846.39	1.62				

Table 2. Results of the analysis of variance for developer productivity. Regression coefficients are in the right column. Adjusted-$R^2 = 0.43$. The largest absolute correlation among predictors was less than 0.28.

of departures but only on the departures of the most experienced developers.

As proposed in Proposition 3, the number of newcomers decreases the productivity of the existing developers and according to Proposition 4, new developers do not reach full productivity immediately. However, the regression coefficient is not significantly different from zero (p-value $= 0.16$). Perhaps some of the new members increase the productivity according to Proposition 2, thus moderating the negative impact caused by the need for training.

The proximity to prior and subsequent organizational change events both have highly significant F statistics (p-value < 0.01) and the estimated regression coefficients are significantly different from zero (p-value $< 10^{-10}$). However, the proximity to a subsequent reorganization explains five times more of the variance than the proximity to a past reorganization. This appears to support Propositions 6. Furthermore, the proximity to the forthcoming organizational change has more pronounced effect than the proximity to the past change.

Though not related to our propositions, the number of employees supervised by a developer has an intuitive effect of reducing developer productivity, perhaps by requiring to spend more time on activities related to supervising.

7. Threats to validity

While we considered the change of the organization ID for a developer and arrival and departure of other emploeeys from that organization, there are many other types of organizational changes that may affect developer productivity. In particular, we do not distinguish between a developer moving to another project versus the organization changing its ID.

Two-month period we considered for the purpose of arrival and departure of emploeeys, may not be sufficient to assess the impact of departures as they may manifest over longer periods of time.

The absolute value of correlations among predictors in the model did not exceed 0.27, thus collinearity effects are not likely.

8. Conclusions

The organization volatility as measured by the number of new developers and proximity to an organizational change was observed to reduce developer productivity. Furthermore, the effect of the arrival of new developers is not as pronounced as the effects of the proximity to the organizational change.

The analysis we conducted could not confirm the impact of the potential gaps in tacit knowledge left by departing developers, perhaps because these gaps need more time to manifest themselves, or may affect the quality of the software more than productivity.

In this initial analysis we did not attempt to identify which organizational changes moved the organization away from the product structure and which changes moved it closer to the product structure. However, all organizational changes appear to decrease developer productivity temporarily, especially prior to the actual change.

While the negative impact on productivity has been established in organizational downsizing situations, we observe it in a more general organizational change scenarios. Unlike prior studies we directly and continuously observe employee (developer) output rather than having discrete, self-reported, and partial (due to non-response) results of surveys. We also introduce a number of measures for organizational change based on reporting structure and find proximity to the subsequent organizational change to be the most important driver of productivity reductions.

More detailed investigations of the nature of organizational change and the nature of productivity reductions observed in this study are likely to provide practical recommendations on ways to organize and reorganize software development.

We expect this finding to add an important piece to a puzzle involving the understanding of how the dynamic relationship between the product and the organization affect key software engineering outcomes.

References

[1] A. J. Albrecht and J. R. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Trans. on Software Engineering*, 9(6):638–648, 1983.

[2] T. Amabile and R. Conti. Changes in the work environment for creativity during downsizing. *Academy of Manegement Journal*, 42:630–640, 1999.

[3] M. Armstrong-Stassen. Coping with downsizing: A comparison of executive-level and middle managers. *International Journal of Stress Management*, 12(2):117–141, 2005.

[4] D. Atkins, T. Ball, T. Graves, and A. Mockus. Using version control data to evaluate the impact of software tools: A case study of the version editor. *IEEE Transactions on Software Engineering*, 28(7):625–637, July 2002.

[5] J. Barron, M. Hannan, and M. Burton. Labor pains: organizational change and employee turnover in young, high-tech firms. *American Journal of Sociology 106 (January): 9601012*, 106(4):960–1012, January 2001.

[6] V. Basili and R. Reiter. An investigation of human factors in software development. *IEEE Computer*, 12(12):21–38, December 1979.

[7] C. Brotheridge and A. Grandey. Emotional labor and burnout: Comparing two perspectives of ”people work”. *Journal of Vocational Behaviour*, 60:17–39, 2002.

[8] C. Brotheridge and R. Lee. Testing a conservation of resources model of the dynamics of emotional labor. *Journal of occupational Health Psychology*, 7:57–67, 2002.

[9] D. J. Cohen and L. Prusak. *In Good Company: How Social Capital Makes Organizations Work*. Harward Business Press, 2001.

[10] M. E. Conway. How do committees invent? *Datamation*, 14(4):28–31, April 1968.

[11] B. Curtis. Substantiating programmer variability. In *Proceedings of the IEEE 69*, July 1981.

[12] E. Geisler. Organizational change phenomena, managerial cognition, and archival measures: Reconceptualization and new empirical evidence. Technical Report 99-02, Stuart School of Business, Illinois Institute of Technology, August 1999.

[13] J. D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally-distributed software development. *IEEE Transactions on Software Engineering*, 29(6):481–494, June 2003.

[14] A. Hochschild. *The managed heart: The commrecialization of feeling*. University of California Press, Brekeley, CA, 1983.

[15] A. Mockus. Succession: Measuring transfer of code and developer productivity. In *2009 International Conference on Software Engineering*, Vancouver, CA, May 12–22 2009. ACM Press. To appear.

[16] A. Mockus and J. Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *2002 International Conference on Software Engineering*, pages 503–512, Orlando, Florida, May 19-25 2002. ACM Press.

[17] A. Mockus and D. Weiss. Interval quality: Relating customer-perceived quality to process quality. In *2008 International Conference on Software Engineering*, pages 733–740, Leipzig, Germany, May 10–18 2008. ACM Press.

[18] A. Mockus and D. M. Weiss. Globalization by chunking: a quantitative approach. *IEEE Software*, 18(2):30–37, March 2001.

[19] I. Nonaka. A dynamic theory of organizational knwledge creation. *Organizational Science*, 5(1):14–37, February 1994.

[20] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[21] J. Van Maanen and E. Schein. Towards a theory of organizational socialization. In B. Staw, editor, *Research in organizational behavior*, volume 1, pages 209–264. JAI Press, Greenwich, CT, 1979.

[22] F. W. T. F. W. *The Principles of Scientific Management*. Harper & Brothers, 1911.

[23] E. Webb, D. Campbel, R. Schwartz, and L. Sechrest. *Unobtrusive Measures: Nonreactive Research in the Social Sciences*. Rand McNally Colledge Publishing Company, Chicago, IL, 1966.

[24] M. Zhou and A. Mockus. Learning in offshored and legacy software projects: How product structure shapes organization. In *ICSE Workshop on Socio-Technical Congruence*, Vancouvre, Canada, May 2009.