

Factors Leading to Integration Failures in Global Feature-Oriented Development: An Empirical Analysis

¹E.Geetha Rani, ²CH.Suguna Latha, ³D.Anusha, ⁴M.Satya Sukumar, ⁵T.Swathi

¹Dept. of Information Technology, Gudlavalleru Engineering College, Gudlavalleru, JNTUK, AP, India

^{2,5}Dept. of Information Technology, Gudlavalleru Engineering College, Gudlavalleru, JNTUK, AP, India

³Dept. of Information Technology, PVPSIT, Vijayawada, JNTUK, AP, India

⁴Dept. of Information Technology, PPDCE, Surampalli (V), JNTUK, AP, India

Abstract

Feature-driven software development is a novel approach that has grown in popularity over the past decade. Researchers and practitioners alike have argued that numerous benefits could be garnered from adopting a feature-driven development approach. However, those persuasive arguments have not been matched with supporting empirical evidence. Moreover, developing software systems around features involves new technical and organizational elements that could have significant implications for outcomes such as software quality. This paper presents an empirical analysis of a large-scale project that implemented 1195 features in a software system. We examined the impact that technical attributes of product features, attributes of the feature teams and cross feature interactions have on software integration failures. Our results show that technical factors such as the nature of component dependencies and organizational factors such as the geographic dispersion of the feature teams and the role of the feature owners had complementary impact suggesting their independent and important role in terms of software quality. Furthermore, our analyses revealed that cross-feature interactions, measured as the number of architectural dependencies between two product features, are a major driver of integration failures. The research and practical implications of our results are discussed.

Keywords

Management, Measurement, Human Factors.
Metrics – Process Metrics, Product Metrics.
Management – Life Cycle, Programming Teams.
Feature-Oriented Development, Cross-Feature Interaction, Global Software Development.

I. Introduction

A product feature is an important concept in software development because it conveys attributes of a product relevant to all the stakeholders involved in the development process. For example, a customer might communicate his/her requirements for a particular product as a collection of characteristics or features the product ought to have or a software architect might make design decisions based on the collection of features that will be part of a product. Over the years, researchers and practitioners alike have articulated a number of benefits that could be garnered from adopting a feature-driven development approach. For instance from a technical point of view, past work has argued that feature-driven development enhances development flexibility (e.g. [28]), facilitates formal modeling of systems (e.g. [31]) and even leads to higher levels of quality (e.g. [27]). From a process perspective, the concept of features is an integral part of the software product lines approach [12]. Finally from an organizational perspective, researchers have argued that features represent very valuable entities that can facilitate coordination, collaboration and overall governance of software projects [8, 32]. Despite the growing popularity and

adoption of feature-driven software development, we have very limited understanding as to how the technical attributes of features impact traditional development outcomes such as productivity and quality. Furthermore, there are also a number of organizational parameters that are involved in the usage of a feature-driven development approach such as configuring feature teams and selecting feature owners. The current state of the art on how those organizational aspects of feature-oriented development impact outcomes such as development productivity or software quality consists mostly of anecdotal evidence. In this paper, we examine the impact that technical attributes of product features, attributes of the feature teams and cross-feature interactions have on integration failures. We collected data from a large-scale global software development project that implemented 1195 features over a period of 32 months of activity. Our results show that technical and organizational factors have complementary impact suggesting their independent and important role in terms of software quality. In particular, we found that the level of technical coupling within features, the concentration of that coupling across architectural components, the geographic distribution of feature teams as well as the group membership of feature owners were important factors leading to integration failures. Furthermore, our analyses revealed that cross-feature interactions, measured as the number of architectural dependencies between two product features, are a major driver of integration failures. The rest of the document is organized as follows. We first discuss past work and the research questions examined in this paper. Then, we describe our research setting, research design and results. We conclude with a discussion of the contributions, limitations and future research directions. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post servers or to redistribute to lists, requires prior specific permission and/or a fee. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and those copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

II. Integration Failures in Feature-Oriented Development

Software quality has been the subject of a large body of past research work, and numerous factors that negatively impact software quality have been identified (e.g. [7, 10, 13-14, 17, 25-26, 30, 34]). A distinctive characteristic of that work is the focus on particular units of a software system such as files, classes, modules, components or binaries and how their technical attributes

and patterns of change impact their quality. Those software entities represent tangible boundaries that support critical design principles such as abstraction, information hiding and, in general, modular design. Feature-driven development introduces a distinctive new element, the product feature, which has characteristics that differ from what we traditionally consider as a software entity (e.g. a file, class, module or component). First, product features, like aspects [20] tend to cut across those traditional software entities. In doing so, they define new technical boundaries which might include, for instance, entire components or portions of them. Second, those new boundaries tend to create a challenging tension between adequately designing and implementing the necessary functionality of the product features and the integrity of the architectural components or modules involved or impacted by the various product features. Given these differences between features and the traditional software entities, it is important to consider how the specific technical attributes of features such as the characteristics of architectural dependencies embedded in the feature impact software quality. In particular, we are interested in software quality outcomes in the context of integrating the various parts that constitute a feature, which is a critical process step in feature-driven development. This leads to the following research question:

A. RQ1a: What is the Impact of Technical Attributes of a Product Feature on Failures During the Integration of that Feature?

It is well established that software development involves not just a technical dimension but also a socio-organizational one. For example, individual-level experience, either technical or domain specific, has been found to be an important factor leading to errors and failures in the development of software systems (e.g. [4, 13-14]). Configuration aspects of the development teams are also another important set of factors that impact important software development outcomes such as quality. For instance, the geographic dispersion of the team members (e.g. [9-10]), the characteristics of the leaders and manager as well as their leadership styles [33], the patterns of interaction on task tracking systems [34] as well as the number of individuals involved in the development of a piece of software (e.g. [17]) impact software quality. A key aspect of feature-driven development is the establishment of feature teams to develop the necessary features [27]. Those teams tend to be short-lived (just for the duration of developing a feature) and individuals tend to be members of multiple feature teams. Socio-organizational factors seem likely to play a major role in the quality outcomes of feature teams, but given the substantial differences in development organized around feature teams as opposed to traditional teams, it is difficult to know what attributes will have important effects. This leads to the following research question:

B. RQ1b: What is the Impact of Organizational Attributes of the Feature Team on Failures During the Integration of a Product Feature?

The previous paragraphs have considered technical and organizational factors in the context of a single product feature. However, product features do not exist in isolation. They depend or interact with other product features. A growing body of work has shown that cross-features interactions are a major challenge. Calder and colleagues [6], argued that current research challenges include understanding where potential interactions arise, how to determine that an interaction did in fact occur and how to resolve it. Those gaps in the literature lead us to the following research question:

C. RQ2: How can we assess cross-feature interactions and what is their impact on failures during the integration of product features?

III. What Factors Drive Failures When Integrating Features?

The first step in our investigation was to examine how technical attributes of a product feature as well as characteristics of the feature teams responsible for developing that product feature impact the occurrence of software failures at the time of integrating the product feature. The literature on software failures is vast and over the years numerous aspects of software systems as well as aspects of the development process have been linked to failures. That work guided us in the selection of independent variables as well as control factors that might impact failures during the integration of features. The rest of this section describes in detail the measures and statistical models used in our analyses followed by our results.

A. Description of the Measures

In order to address research questions 1a and 1b, we collected a number of measures from various data sources including the project's software repositories and documentation as well as human resource records.

1. Measuring Integration Failures

As discussed earlier, the I&T team integrated each feature individually. That process allowed the team to run a collection of integration tests to evaluate the feature just integrated as well as all the previously integrated features. The I&T team recorded for each integration test whether the tests associated with the recently Integrated feature passed or not. They also recorded whether the tests associated with previously integrated features passed or not. Our outcome measure is a dichotomous variable where a 1 indicates that at least one of the tests performed by the I&T team at the time of integrating a feature failed. Otherwise, the variable is set to 0.

2. Additional Control Factors

We collected a number of additional measures that past research has shown to be related to software quality and, consequently, relevant to modeling integration failures [2, 5, 17, 26, 30]. For each product feature, we collected the Number of Modification Requests as reported in the modification request (MR) tracking system associated with each feature. We also collected the Total Size in LOCs of the components that were involved in a feature. It is well established that over time development organizations learn and mature their processes and practices and consequently, tend to reduce mistakes and errors. Therefore, we computed the variable Time that represents the week within the project on which the feature was integrated. Finally, we collected measures of experience based on the approaches used by Boh and colleagues [4] and Espinosa and colleagues [15], which utilize the data in software repositories as the basis for assessing experience. We measured prior experience on the product in two different ways. Average MR Experience assessed the average number of MR that the feature team member worked on prior to the focal feature. Average Component Experience measured the average number of times that the feature team members modified the architectural components associated with the focal feature prior to the beginning of the development of the feature.

B. Description of the Model

Our dependent measure is a dichotomous variable. Consequently, we used logistic regression models to examine research questions 1a and 1b. We followed a traditional hierarchical approach where we start our analyses with a baseline model that contains only control factors. In subsequent models, we added the various independent measures associated with the different research questions. This modeling approach allows us to understand the independent and relative impact on integration failures of each set of factors. In order to assess the fit of each model, we report the deviance of each model as well as the percentage of deviance explained by the model. The deviance of a model is defined as “ $-2 * \log\text{-likelihood}$ of the model” and lower values are associated with better fit of the model to the data. The percentage of the deviance explained is a ratio of the deviance of the null model (contains only the intercept) and the deviance of the final model. In order to simplify the interpretation of the results, we report the odds ratios associated with each measure instead of reporting the regression coefficients. Odds ratios larger than 1 indicate a positive relationship between the independent and dependent variables whereas an odds ratio less than 1 indicates a negative relationship.

C. Results

1. Preliminary Analysis

The first step in our analysis consisted in examining various descriptive statistics of the measures described earlier. Several variables had skewed distributions so they were log-transformed. In the next step, we performed various collinearity diagnostics. A Variance Inflation Factors (VIF) analysis revealed that several of the measures were highly collinear. In accordance with well established recommendations, we removed from our analyses all the variables with VIF values above 5 [23]. A pair-wise correlation analysis among those remaining measures showed levels of correlation that were not problematic with the highest values being 0.359 and 0.258 between Changed LOCs and Number of Dependencies and Time, respectively. Table 1, reports the results of our regression analyses examining the impact on integration failures of various technical and organizational factors. Model I reports the odds ratios associated with the control factors included in our analyses. As expected, Time and higher levels of Average Component Experiences are associated with lower probability of failures. For instance, an additional week in the development project corresponding to a unit increase in the variable Time reduces the likelihood of failure (odds ratio equal to 0.992 – lower than 1) by 0.8% considering all other factors constant.

2. The Impact of Technical Attributes of Features

We examined research question 1(a) with model II in Table 1. The model includes the technical attributes of the feature to study their impact on integration failures. Our results do not provide evidence that either the amount of source code changed (Changed LOCs) during the development of the feature or the concentration of those changes across the various architectural components affected by those changes (Concentration of Changed LOCs) impacted the likelihood of integration failures. On the other hand, the level of technical coupling (Number of Dependencies) among the components involved in the feature and the degree of concentration of that coupling (Concentration of Number of Dependencies) do have a statistically significant effect on integration failures. We observe that the higher the number of architectural dependencies among the components that are impacted by a feature, the higher

the likelihood of failures (odds ratio > 1). Moreover, the higher the concentration of the coupling in a smaller number of components is, the higher is the likelihood of integration failures.

Table 1: Odds Ratios from Regression Assessing Factors Driving Feature Integration Failures

	Model I	Model II	Model III	Model IV
Time	0.992*	0.990*	0.990*	0.989*
Average Component Experience (log)	0.487*	0.984+	0.741+	0.754
Changed LOCs		1.021	1.009	1.063
Concentration of Changed LOCs		1.045	1.028	1.036
Number of Dependencies (log)		1.107*	1.091*	1.091*
Concentration of Number of Dependencies		1.032**	1.046**	1.078**
Number of Groups			1.101*	1.051*
GSD			13.924**	14.964**
Feature Owner Belongs to Highly Coupled Component			0.789	0.396
Feature Owner Belongs to Highly Coupled Component			0.839**	0.819**
Concentration of Changed LOCs X F. Owner Belongs to Highly Coupled Component				1.032
Concentration of Number of Dependencies X F. Owner Belongs to Highly Coupled Comp.				0.977**
GSD X Feature Owner Belongs to Highly Coupled Component				3.736
GSD X Feature Owner Belongs to Highly Coupled Component				0.926
Deviance of the Model	755.2	639.0	438.4	412.2
Deviance Explained	11.7%	25.3%	46.4%	51.8%

(+p < 0.1; *p < 0.05; **p < 0.001)

3. Additional Analysis

We performed additional analyses to examine the potential conditional impact of particular factors such as which group the feature owner belonged to and the geographic dispersion of the feature team. The conditional impact of the variables can be studied with interaction terms in a regression model. An interaction between two factors, for instance, GSD and Feature Owner Belongs to Highly Coupled Component, allows us to examine how the impact of one factor (e.g. Feature Owner Belongs to Highly Coupled Component) on the dependent variable changes for different values of the second factor (e.g. feature team is collocated or not). In model III, we found that the group to which the feature owner belongs is an important organizational factor in the context of integration failures. However, the impact of such factor might differ depending whether the feature team members are geographically distributed or not. In addition, the level of concentration in the changes to the code or in the technical coupling among components might moderate the impact of having the feature owner belonging to a particular group. For example, it might be only beneficial to have the feature owner belong to the group that is responsible for the component with the highest amount of changes or dependencies only when the concentration levels are relatively high. In order to explore those potential conditional effects, model IV of Table 1, includes several interaction terms that were selected based on the results reported in models II and III. All independent variables were meancentered, an approach traditionally used to address the collinearity issues introduced by the interaction terms.

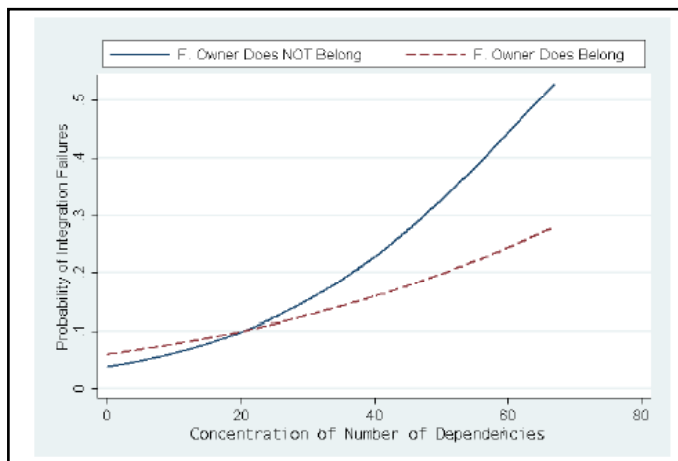


Fig. 1: The Interplay Between the Concentrations of Technical Coupling within a Feature and the Feature Owner Belonging to the Highly Coupled Component

The results reported in model IV show only one interaction term, Concentration of Number of Dependencies X Feature Owner Belongs to Highly Coupled Component, is statistically significant suggesting that the impact of the concentration of the technical coupling within a feature is moderated by whether the feature owner belongs to the group responsible for the highly coupled components or not. An odds ratio below 1 indicates that when the feature owner belongs to the group responsible for the highest coupled component, the negative impact of higher levels of concentration of the technical coupling is reduced. Figure 1 illustrates this point by depicting how the probabilities of integration failure estimated by our logistic regression models change as the number of dependencies changes. As the concentration of the number of dependencies increases (x axis), we observe that the estimated probability of failure (y axis) increases faster when the feature owner does not belong to the group responsible for the component with the highest level of coupling (solid line in figure 1) than when the feature owner belongs to the group responsible for the component with the highest level of coupling (dashed line in figure 1). In fact, the probability of failure reduces by half in the latter case where the number of dependencies is higher than 40. Furthermore, we see that this relationship inverts for lower values of concentration in the number of dependencies (values < 20).

IV. Cross-Feature Interactions and Integration Failures

We now turn our attention to cross-feature interactions and their implication for integration of features. In particular, we examine how architectural dependencies that represent relationships between product features impact the occurrence of integration failures in a feature-driven development setting. The rest of the section describes in detail the measures and statistical models used in our analyses as well as the results of our investigations.

A. Description of the Measures

In order to address research question 2, we collected a number of measures from various data sources including the project's software repositories, documentation and human resource records.

1. Measuring Integration Failures

In this case, our unit of analysis is the pair of product features. As described in section 3, product features were integrated into a prerelease development stream individually. Then, the I&T team ran a collection of integration tests every time a feature was

integrated. Using such information, we constructed our dataset of pairs of features in the following way. When a feature F_n was integrated, we created $n-1$ pairs $(F_1, F_n) \dots (F_{n-1}, F_n)$. Considering all 1,195 features developed in the project and integrated, we had a total of 713,416 possible pairs of features. Our dependent measure is a dichotomous variable where a 1 associated with a pair of features (F_x, F_y) indicates that the integration tests associated with either features F_x or F_y failed at the time of integrating feature F_y (assuming that feature F_y was integrated after feature F_x).

2. Measuring Cross-Features

Interactions We measured the Number of Cross-Features Dependencies for a pair of features (F_x, F_y) as the number of architectural dependencies that the components involved in feature F_x had with the components involved in feature F_y . The data about architectural dependencies were extracted from the project's software architecture documentation that contained detailed descriptions of all 107 architectural components and their relationships.

3. Additional Control Factors

We collected a number of additional measures for each pair of features. Since our unit of analysis is the pair of features, integration failures could be impacted by a buggy feature that was integrated in the past rather than by the feature being integrated. In order to control for this effect, we constructed dichotomous variables, Past Failures in the Past X Weeks. These variables measured the impact of past failures associated with the features that were integrated in the past feature F_x in a given pair (F_x, F_y) when integrating a new feature F_y . We considered 1 to 10 weeks time periods. For example, the variable Past Failures in the Past 5 Weeks would be set to 1 for the pair of features (F_x, F_y) if there has been an integration testing failure associated with F_x in the past 5 weeks prior to integrating feature F_y . We measured the Changed LOCs as the number of non-comment non-empty lines of code that were added, deleted and modified as part of developing both features in each pair.

B. Description of the Model

As in the case of the analyses reported in the previous section, our dependent measure is also dichotomous variable. Consequently, we followed the same modeling strategy (the use of logistic regression models).

C. Results

1. Preliminary Analysis

We performed preliminary analysis similar to those described in earlier section. In order to reduce estimation problems associated

Table 2: Odds Ratios from Regression Assessing the Impact of Cross-Feature Interactions on Integration Failures

	Model I	Model II	Model II
Time	0.981**	0.971**	0.964*
Failures in the Past 5 Weeks	2.127**	1.125*	1.011*
Changed LOCs	1.371**	1.201**	1.203**
Average Component Experience (log)	0.837+	0.997	0.908
Number of Groups	3.006**	4.057**	6.345**
Overlap Among Groups	0.943**	0.919**	0.901**
Same Feature Owner	0.876**	0.871**	0.852**
GSD	4.501**	2.509**	4.895**
Number of Cross-Feature Dependencies (log)		2.911**	4.938**
Number of Groups X Number of Cross-Feature Dependencies			0.607
GSD X Number of Cross-Feature Dependencies			0.799**
Deviance of the Model	12873.9	9413.1	8043.1
Deviance Explained	33.4%	51.3%	58.4%

(*p < 0.1; *p < 0.05; **p < 0.01)

With collinearity, we removed from our analyses all the variables with variance inflation factor values above 5 as suggested by the literature [23]. We also performed a pair-wise correlation analysis among those remaining measures and we did not find any correlations that should be a concern. The highest values were 0.391 and 0.357 between Number of Cross-Feature Dependencies and the variables. Finally in each pair of features (Fx,Fy), past failures associated with a feature Fx prior to the integration of feature Fy was also an important factor leading to integration failures. As discussed earlier, we computed 10 different versions of the variables covering a period of 1 week prior to integration of a feature to 10 weeks prior. We found that the variable corresponding to the 5-week period worked best in our models. We think this particular result is related to characteristics of the project

2. The Impact of Cross-Feature Interactions

Model II introduces the Number of Cross-Feature Dependencies into the analysis and we observe that its impact is statistically significant. The higher the Number of Cross-Feature Dependencies a pair of features have, the higher the likelihood of integration failures to occur. It is also important to highlight that this factor has a major impact in the explanatory power of the model representing 17.9% of the deviance explained by the model. This result suggests that our metric could be a valuable tool for practitioner to identify where potential coordination problems might occur.

V. Future Research

The results of our study have important implications for research. First, our analyses revealed that the measure of cross-feature interaction based on architectural dependency information was a major predictor of integration failures. However, as discussed earlier, such a measure represents a subset of all potential interactions between product features that might exist in a system. Past research has shown that version control data can be used to identify logical dependencies between parts of a software system. Those dependencies have been found to drive relevant coordination needs among developers that when satisfied development productivity improves and probability of software failures to occur is reduced [11]. Then, a potentially valuable future research path is to explore combining those two approaches to measure cross-feature interactions, assess their impact on software quality as well as examine their implications for evolving software system either by adding more features or modifying existing ones. Second, our findings showed an important detrimental impact of geographic distribution on integration failures. Recent work in the areas of software quality and distributed development has shown mixed results. For example, amasubbu and Balan [29], found no evidence that geographic dispersion impacted quality outcomes

of software development projects. Bird and colleagues [3], found that collocated and distributed teams developed binaries with very similar levels of quality. In contrast, Cataldo and Nambiar [9-10] found that different dimensions of geographic dispersion impacted negatively the quality of architectural components as well as the quality outcomes of projects. Such disparity in results might be indicative that the relationship between distributed development and software quality could be moderated by different technical and organizational factors that future research should examine. The relative role of technical and organizational factors associated with product features suggested by our results point out that future research in the area of collaborative technologies to support the development organization could benefit from considering product features as first order entities. Recently, researchers have argued that the concept of a product feature could represent the “glue” to facilitate coordination, collaboration and overall governance of software development endeavors [8, 32]. Our results provide guidance on the set of factors and metrics those collaborative technologies can focus on in order to enhance the coordinative and collaborative capabilities of software development organizations, particularly, those that are geographically distributed. For example, considering product features as a first order entity, a collaborative tool could collect cross-feature dependency information and Automatically suggests potential interdependent engineers based on the individuals’ current work activities and the degree of cross feature Interaction that between the features those individuals are Working on. Finally, the results of our analyses also call for a more systematic evaluation of the range of organizational and governance principles and practices that apply in the context of feature-driven development. Our analyses focused on only three aspects: geographic dispersion, selection of the feature owner and overlap in feature team membership. Certainly there are several other aspects that deserve further investigation such as different configurational Properties of geographic dispersion and the integration of feature oriented development with agile practices to name a few.

References

- [1] Austin, R.D., "The Effects of Time Pressure on Quality in Software Development: An Agency Model", Management Science, 12, 2, 2001.
- [2] Basili, V.R., Perricone, B.T., "Software Errors and Complexity: An Empirical Investigation", Communications of the ACM, 12 (1994), 42-52, 1994.
- [3] Bird, C. et al., "Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista", In Proceedings of the International Conference on Software Engineering (Vancouver, Canada). ICSE'09, 2009.
- [4] Boh, W.F. et al., "Learning from Experience in Software Development: A Multilevel Analysis", Management Science, 53, 8, pp. 1315-1331, 2007
- [5] Briand, L.C. et al., "Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems", The J. of Systems and Software, 51, pp. 245-273, 2000.
- [6] Calder, M. et al., "Feature Interaction: A Critical Review and Considered Forecast. Computer Networks, 41, pp. 115-141, 2003.

- [7] Cataldo, M., "Sources of Errors in Distributed Development Projects: Implications for Collaborative Tools", In Proceedings of the Conference on Computer Supported Cooperative Work (Savannah, Georgia). CSCW'10, 2010.
- [8] Cataldo, M., Herbsleb, J.D., "End-to-end Features and Meta-entities for Enabling Coordination in Geographically Distributed Software Development", In Proceedings of the 2nd International Workshop on Software Development Governance (Vancouver, Canada, 2009) SDG'09, 2009.
- [9] Cataldo, M., Nambiar, S., "On the Relationship Between Process Maturity and Geographic Distribution: an Empirical Analysis of their Impact on Software Quality", In Proc. of the International Conf. on Foundations on Software Engineering (Amsterdam, The Netherlands). FSE'09, 2009.
- [10] Cataldo, M., Nambiar, S., "The Impact of Geographic Distribution and the Nature of Technical Coupling on the Quality of Global Software Development Projects", Forthcoming in J. of Softw. Maint. Evol.: Res. and Pract, 2010.
- [11] Cataldo, M., Herbsleb, J.D., "Coordination Breakdowns and their Impact of Development Productivity and Software Failures", Technical Report CMU-ISR-10-104, School of Computer Science, Carnegie Mellon Univ., 2010.
- [12] Clements, P., Northrop, L., "Software Product Lines: Practices and Patterns", Addison-Wesley, New York, NY, 2002.
- [13] Curtis, B., "Human Factors in Software Development", Ed. by Curtis, B., IEEE Computer Society, 1981.
- [14] Curtis, B., et al., "A field study of software design process for large systems", Comm. of the ACM, 31, 1988.
- [15] Espinosa, J.A. et al., "Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development", Org. Science, 18, 4, pp. 613-630, 2007
- [16] Fitzpatrick, G. et al., "CVS Integration with Notification and Chat: Lightweight Software Team Collaboration", In Proceedings of the Conference on Computer supported Cooperative Work (Banff, Canada). CSCW'06, 2006.
- [17] Graves, T.L. et al., "Predicting Fault Incidence Using Software Change History", IEEE Transactions on Software Engineering, 26 (2000), pp. 653-661, 2000
- [18] Hahn, Y. et al., "Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties", Information Systems Research, 19, 3 pp. 269-391, 2008.
- [19] Harter, D.E. et al., "Effects of Process maturity on Quality, Cycle Time, and Effort on Software Product Development", Management Science, 46, 4 (Apr. 2000), pp. 451-466, 2000.
- [20] Kiczales, G. et al., "Aspect-Oriented Programming", In Proceedings of the European Conference on Object-Oriented Programming (Finland). ECOOP'97, 1997.