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

Marcelo Cataldo
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA 15213
mcataldo@cs.cmu.edu

James D. Herbsleb
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA 15213
jdh@cs.cmu.edu

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.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *process metrics, product metrics.* D.2.9 [**Software Engineering**]: Management – *life cycle, programming teams.*

General Terms

Management, Measurement, Human Factors.

Keywords

Feature-oriented development, cross-feature interaction, global software development.

1. 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.

ICSE'11, May 21–28, 2011, Waikiki, Honolulu, HI, USA

2. 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:

> *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:

> *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:

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

3. RESEARCH SETTING

We collected data from a development organization responsible for producing a navigation system for automobiles. Our data covered 32 months of development activity between 2006 and 2008 corresponding to the last version of the product. One hundred and seventy nine engineers located in 6 development sites distributed across North America, Europe and India participated in the project. Those developers were organized in 13 development teams. Most of the teams involved engineers from at least 2 development sites. Telephone, conference calls and email were the primary mechanisms of communication among distributed teams and engineers. The use of lightweight collaboration technologies such as instant messaging and wikis was not allowed. The project also had daily status meetings organized like scrum meetings.

The system was composed of about 1.5 million lines of code distributed in 6,789 source code files and 107 architectural components. The development responsibilities of each component were assigned to a single development team. The source code files were written mostly in C++ but contained a significant amount of code written in C and Assembly language. All developers had full access to the version control system, the task tracking system and a document repository that contained requirements, architectural and design specifications.

The project involved the development of 1,195 product features. The organization utilized a feature-oriented development approach and it started using that approach 5 years prior to the time covered by our data. In this project, the software architects and the software architecture of the system played an important role in the feature-oriented development process. Software architects were responsible for analyzing the descriptions of each product feature and determining the set of architectural components that would have to be altered or enhanced in order to implement the requested feature. The software architects, then, produce a feature development specification that contained a description of the work to be done in each architectural component. The feature teams used those specifications to do the detailed design and implementation of the feature. Once a feature development specification was written, the management of the project defined a feature team that would be assigned to that particular product feature. The feature teams were composed of engineers for the teams responsible from the architectural components that were impacted by the feature as indicated in the feature specification document. Management also assigned a feature owner to each product feature. That individual was a senior engineer, a group leader or a manager from one of the teams involved in the feature team. The organization did not have any specific process for selecting feature owners. However, project executives tended to base their selection on

experience with the type of product feature and they also tried to balance the feature ownership load across all feature owners.

Once completed, each product feature was delivered to an integration and testing (I&T) team who was responsible for two activities. First, they merged the source code of the feature into a pre-release branch in the version control system. Following that step, they ran a collection of integration testing suites that evaluated the feature just integrated as well as all the previously integrated features. The test suite consisted of a collection of tests defined by architects and engineers to evaluate the requirements of each feature. The I&T team implemented the test suites in two complementary parts. One part consisted of simple tests (similar in nature to a "smoke test") that were run on the software system using a simulator for the real hardware. A second part of the testing included the full implementation of the tests and it was executed against the software system running on the actual hardware of the embedded system. The outcomes of the integration tests were recorded and kept as part of the project's repositories. 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 features integrated in the past passed or not.

Our empirical analyses are organized in two parts. In section 4, we examine research questions 1a and 1b related to technical and organizational attributes of features that lead to integration failures. In section 5, we examine the role of cross-feature interactions on integration failures (RQ2).

4. 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 a software systems as well as aspects of the development process have been linked to failures (e.g. [1, 5, 9, 17, 19, 22, 26, 30, 34]). 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.

4.1 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.

4.1.1 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.

4.1.2 Independent Variables

Our independent measures are organized in two groups. We first describe a collection of technical attributes of the features that were assessed. The development of each product feature consisted of one or more transactions in the version control system that impacted one or more architectural components. From those transactions, we collected the *Changed LOCs*, which accounts for the number of non-comment non-empty lines of code that were added, deleted and modified as part of developing a feature. In addition to the amount of change, the locality of those changes might be relevant to integration failures. For example, if the majority of the changes are collocated in the single component involved in a feature, it might be easier to test them and assess their potential impact and implications. On the other hand, if the changes are dispersed across all the components involved in the feature, testing the changes or understanding their potential implication might be more challenging. In order to reach a better understanding of how those different patterns of code change within a feature might impact integration failures, we computed the *Concentration of Changed LOCs* measure. Building on past work that examined the notion of dispersion of software engineers across development locations (e.g. [7] and [10]), we evaluated two approaches for constructing the measure. One approach (based on [7]) uses the standard deviation of the changes made to each component involved in a feature. Smaller values indicate that the changes are relatively evenly distributed among the changed components. A second approach builds on entropy-based measures (e.g. [10]) to assess the degree of concentration of changes in one particular architectural component. Both measures had equivalent impact of the results from the regression models. We chose to include the measure based on the first approach because the fit of the regression models was higher than the fit obtained when using the entropy-based approach.

Technical coupling among constituent parts of a software system has consistently been shown to impact software failures (e.g. [5, 9, 26, 30]). A measure of the extent of the technical coupling within each feature was collected from the architectural description of the system. The *Number of Dependencies* variable refers to the number of interfaces that the architectural components involved within a feature had with each other. In other words, *Number of Dependencies* measures the inter-component coupling within the boundaries of a product feature. As in the case of changes in the source code, the locality of the technical dependencies within a feature might impact failures. Past work in the area of coupling and cohesion (e.g. [9, 26, 30]) would suggest that if the inter-component dependencies within a feature are concentrated in one or few components, their high levels of coupling could lead to failures. However, a product feature aggregates those different patterns of technical coupling, therefore it is unclear how they might impact failures when considering features as the unit of analyses rather than architectural components. In order to shed light on the relationship between technical coupling and product features, we computed the *Concentration of Dependencies* variable as the standard deviation of the number of dependencies among all possible pairs of components involved in a feature. Smaller values indicate that the dependencies are relatively evenly distributed among the components involved in a product feature. As indicated in the previous paragraph, this approach of constructing the measure proved to be a better measure than other approaches (e.g. an entropy-based measure) for the statistical models used in our analyses.

In a second set of measures we assessed organizational properties of the feature teams involved in the development of a feature. *GSD* is a dichotomous variable where 1 indicates that the feature team members were located in different development sites; otherwise it is set to 0. It is worth pointing out that all the project members that were collocated work in a single open space in individual cubicles. We also measured the *Number of Developers* that were part of the feature teams. As discussed earlier, feature team members were selected from formal groups responsible for the various architectural components. Therefore, we also measured the *Number of Groups* involved in the development of each feature. Finally, we constructed two measures that assess the relationship between the organizational and the technical dimensions of developing a product feature. The feature owner has an important role within the development effort associated with a feature because it is the individual responsible for making sure the feature team is able to accomplish its goal of developing the feature. In our research setting, feature owners were selected from among the members of the groups responsible for the components being affected by a product feature suggesting that feature owners might have particular sets of skills, experiences and relationships that might benefit certain members of a feature team and hinder others. Therefore, we computed the *Feature Owner Belongs to Highly Coupled Component* measure as a dichotomous variable, which is set to 1 if the feature owner belonged to the group responsible for the component with the highest number of technical dependencies within the assigned feature. Similarly, we computed the measure *Feature Owner Belongs to Highly Changed Component*, which is set to 1 if the feature owner belonged to the group responsible for the component that was changed the most as part of the development of the focal feature.

4.1.3 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.

4.2 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-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.

4.3 Results

4.3.1 Preliminary Analyses

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.

4.3.2 The Impact of Technical Attributes of Features

We examined research question 1a 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	1.089	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 Changed 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 Changed 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 Changed Component*				3.736
GSD X *Feature Owner Belongs to Highly Coupled Component*				0.926
Deviance of the Model	755.2	639.0	458.4	412.2
Deviance Explained	11.7%	25.3%	46.4%	51.8%

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

4.3.3 The Impact of Organizational Attributes of Feature Teams

In model III, we examined the role that organizational attributes of the feature teams have on integration failures (RQ1b). We observe that several organizational factors have a statistically significant impact on integration failures. As the number of development groups that work on a feature increases, so does the probability of occurrence of integration failures. In addition, geographic dispersion of the feature team has a major impact of integration failures. When the engineers that worked on the feature were geographically distributed, the likelihood of integration failures is almost 14 times higher than when all engineers are in the same location (odds ratio = 13.924). We also examined how the group membership of the feature owner impacted the outcome of the feature teams. We found that selecting the feature owner from the group that is responsible for the architectural component with the highest level of technical coupling within the product feature helps reduce the probability of integration failures to occur. Specifically, having the feature owner belong to the group responsible for the highest coupled component decreases the likelihood of failures by about 20% (odds ratio=0.839) compared to not having the feature owner belong to that development group.

Models II and III provide insight on the relative impact on failures of the technical attributes of product features and organizational characteristics of feature teams. Organizational attributes explain 21.1% of the deviance in the data (the difference in the deviance explained between model II and III) whereas technical attributes of feature explained 13.6% of the deviance. Those results highlight the significant impact that the characteristics of feature teams have on the outcomes of feature-driven development endeavors

4.3.4 Additional Analyses

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 mean-centered, an approach traditionally used to address the collinearity issues introduced by the interaction terms.

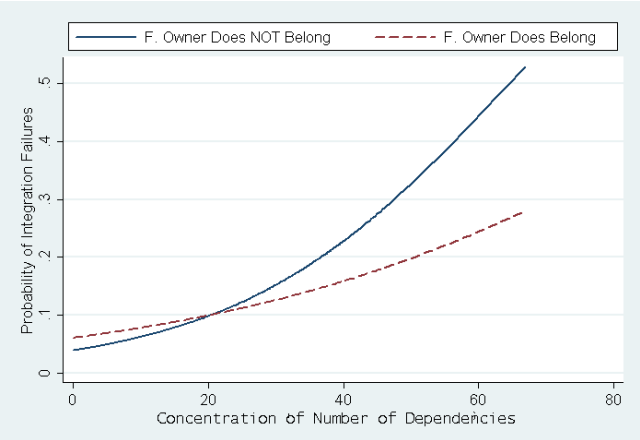

Figure 1: The Interplay between the Concentration of Technical Coupling within a Feature and the Feature Owner Belonging to the Highly Coupled Component.

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).

5. 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.

5.1 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.

5.1.1 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) … (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 measures 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).

5.1.2 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.

5.1.3 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. The measure *Number of Modification Requests* captured the number of MRs associated with each pair of features as reported in the MR tracking system. 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* for each pair of features (F_x, F_y) that represents the week within the project on which the feature F_y was integrated.

We also assessed a set of organizational properties of the feature teams involved in the development of each pair of features. *GSD* is a dichotomous variable where 1 indicates that the members of either feature teams involved in the development of the features in a given pair (F_x, F_y) were located in different development sites; otherwise it is set to 0. As indicated earlier, all the project members that were collocated work in a single open space in individual cubicles. We also measure the *Number of Groups* involved in the development of each pair of features as well as the *Number of Developers* that were part of those feature teams. The engineers involved in the development of a pair of features that belong to the same formal groups responsible for certain architectural components might have more opportunities to interact, communicate and coordinate than if they belong to different formal teams. Then, we computed the *Overlap Among Groups* as the proportion of feature team members from F_x that belong to the same set of formal groups that the members of the feature team associated with F_y. We also computed the *Same Feature Owner* measure as a dichotomous variable set to 1 if the same individual was the feature owner for the pair of features; otherwise the measure was set to 0. Finally, we collected measures of experience based on the approaches used by Boh and colleagues [4] and Espinosa and colleagues [15]. We measured prior experience on the product in two different ways. *Average MR Experience* measures the average number of MR that the feature teams' members associated with a pair of features worked on prior to the focal feature. *Average Component Experience* represents the average number of times that the feature teams' members modified the components associated with each pair of features.

5.2 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) described in section 4.2.

5.3 Results

5.3.1 Preliminary Analysis

We performed preliminary analysis similar to those described in section 4.3.1. 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**	4.037**	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 *Number of Groups* and *Changed LOCs*, respectively.

Table 2 reports the results of our regression analyses examining the impact of cross-features interactions on integration failures. Model I presents the odds ratios associated with the control factors included in our analyses. As expected, the later in the project's lifecycle a feature is integrated (the *Time* variable), the lower the probability of integration failures, a result consistent with learning and development maturity arguments. In addition, higher *Number of Groups* involved in the development each pair of features and having those groups geographically distributed, increased significantly the likelihood of integration failures to occur. On the other hand, two organizational factors, the proportion of members of the feature teams that developed a pair of features overlapped or belonged to the same formal groups increased (the *Overlap Among Groups* variable) and having the same feature owner for both features in the pair (the *Same Feature Owner* variable) had a positive impact of integration failures, reducing their probability of occurrence. Finally in each pair of features (F_x, F_y), past failures associated with a feature F_x prior to the integration of feature F_y 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. An analysis of the defects reports stored in the defect tracking system revealed that the average time to resolve a defect was 11 days and the standard deviation was 18 days, suggesting that in most defects were resolved within a 5-week period.

5.3.2 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.

5.3.3 Additional Analyses

We performed additional analyses to examine whether the strong impact of cross-feature interactions on integration failures was conditional on other factors. As discussed in section 4.3.4, conditional or moderating effects can be analyzed with interaction terms in a regression mode. Model III, we introduced two interaction terms of particular interest: *Number of Groups* X *Number of Cross-Feature Dependencies* and *GSD* X *Number of Cross-Feature Dependencies*. Geographic dispersion and higher number of individuals involved in the development are two well-established factors that increase the coordination complexity of software development endeavors. Since model II showed such a strong negative impact on failures from the *Number of Cross-Feature Dependencies*, it is important two understand if such impact changes as the number of groups involved in the development of a pair of features changes or whether those groups are geographically distributed or not.

The results reported in model III show only one interaction term, *GSD* X *Number of Cross-Feature Dependencies*, is statistically significant. The odds ratio associated with the interaction term is below 1 (0.799) suggesting that the impact of cross-feature dependencies is lower when developers of the feature teams are geographically distributed than when they are collocated. These are unexpected results. Figure 2 illustrates the *GSD* X *Number of Cross-Feature Dependencies* interaction effect by depicting the changes in the estimated probabilities of having integration failures as the number of cross-feature dependencies changes. As the number of cross-feature dependencies increases (x axis), we observe that there is a point (values > 4 in the log-transformed measure on the x axis) at which the probability of integration failures (y axis) increases significantly faster when the features teams that worked on a pair of features are collocated (solid line in figure 2) than when feature teams are geographically distributed (dashed line in figure 2). It is important to point out that for levels of the cross-feature dependencies measure below 5.5 (in the log-transformed values of the measure), the probability of having integration failures is more than double for the GSD case than for the collocated case. Such a result is consistent with the growing body of work showing the detrimental effects of distribution on quality (e.g. [7, 9]). However, when features are very highly interrelated (values > 6), our results show that the impact of cross-feature dependencies is lower when feature teams are geographically distributed. One possible explanation for

this result is that the work practices developed by collocated teams might allow them to handle certain levels of interdependence between features very well. For example, dependencies might be handled more informally because the interdependent engineers are physically collocated. However, beyond a certain point (e.g. in our analysis values > 4), those work practices failed to adequately identify and manage the interdependencies between features. On the other hand, distributed teams are always at a disadvantage and recognizing such condition they might develop different work practices to manage dependencies that help them cope better with high levels of interdependence.

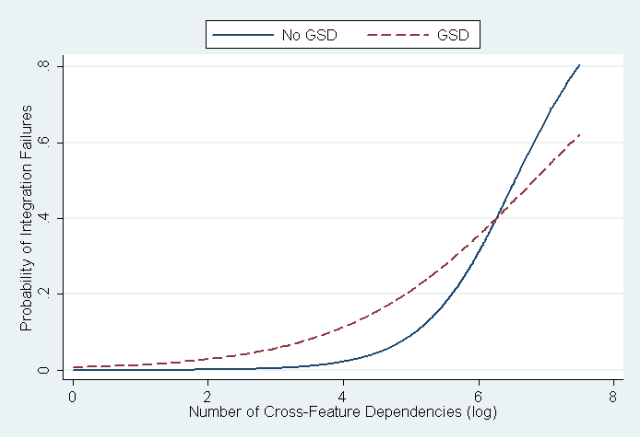

Figure 2. The Interplay between the Geographic Distribution of the Feature Teams and Cross-Feature Dependencies.

5.3.4 Assessing the Robustness of the Results

We performed one final analysis to assess the robustness of the results reported in table 2. Our large dataset is characterized by having a low proportion of 1s in the outcome variable (an integration failure associated with a pair of features), which is known as rare events data [21]. A traditional logistic regression run against a rare events dataset tends to underestimate the probability of the outcome [21]. Given that potential problem, we used a strategy to overcome it as suggested by Hahn and colleagues [18] and, in the process, evaluate the robustness of our results. The approach is known as choice-based sampling and consists in strategically constructing samples from the original dataset based on the values of the outcome variable. We followed this procedure. First, we constructed two dataset with a random sample of 50% of the pairs of features that had integration failures. We then match those pairs with 5 pairs in one dataset and 10 pairs in the other that had the outcome variable 0. These pairs were created using features integrated within a week of the matched pair and involved one of the features in the match pair. We ran the same analyses reported on table 2 on these two additional datasets. The results were all consistent with the ones reported in table 2, providing additional confidence in our results.

6. DISCUSSION

Feature-driven development is a promising approach. In this paper, we set out to empirically study how technical and organizational factors impact outcomes in projects that use a feature-driven development approach in order to further our understanding of its potential. Specifically, we examined the impact that technical attributes of product features and attributes of the feature teams that developed such feature have on one particular dimension of software quality, integration failures. Our results showed the amount of architectural dependencies contained within a feature as well as how those dependencies are distributed across components have an important effect on failures. Specifically, higher levels of technical coupling and higher concentration of such coupling in a small set of architectural components significantly increase the probability of failures at the time of integrating a product feature. Most importantly, 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. We also found that several attributes of the feature teams impacted quality. The number of engineers involved in the development of a feature and their geographic dispersion were detrimental to quality. However, our analyses showed also that selecting a feature owner that is involved with a highly coupled architectural component that is part of a product feature helps overcome the detrimental effects that other technical and organizational factors have on the likelihood of integration failures to occur.

The work reported in this paper has four important contributions to the software engineering literature, in particular, to the work on feature-oriented development. First, our results provide one of the very first empirical evaluations of a feature-oriented development setting and its implications for software quality. Second, our analyses explored how the technical and the organizational dimensions of feature-oriented development impacted integration failures as well as how the interplay between both dimensions impacted such failures. Third, we evaluated an approach to assess the impact of cross-feature interactions and the results showed that our measure based on architectural dependency information was a major driver of integration failures accounting for almost 18% of the deviance in our model. Finally, our results provide concrete guidance to the practice of feature-oriented development. We discuss in detail the pragmatic implications in section 6.3.

6.1 Limitations

Our study has several limitations worth highlighting. First, our work examined a single development organization and a single system developed by that organization, which raises concerns regarding external validity. However, the characteristics of the system (e.g. embedded system developed in a combination of C++, C and assembly programming languages) as well as the processes and work practices used by the studied organization are similar to those found in the telecommunication, healthcare, infotainment and automotive industries. Therefore, we think that our findings are applicable across a wide spectrum of corporate settings that develop embedded systems using feature-oriented development approaches.

As discussed in the literature (e.g. [6]), cross-feature interactions could stem from multiple sources including architectural dependencies, logical or semantic dependencies not adequately represented in architectural descriptions or even from unknown dependencies among different parts of a software system. Our measure of cross-feature interaction is based on architectural dependency information and captures only a fraction of the possible cross-feature interactions that might exist in a system representing a limitation of our analysis. However, the strong impact that our measure had on the probability of integration failures raises a pair of interesting questions: how much additional impact might other types of cross-feature interactions have on failures? And how can we measure such interactions? We elaborate on these issues further in the implications for future research section.

Third, the organization we studied did not allow the use of lightweight collaboration technologies such as instant messaging and wikis. Past research have found that such technologies help software engineers be more aware of the activities that are taking place in the project and coordinate and collaborate better (e.g. [16, 24, 34]). It is possible that the impact of some of the factors considered in our analyses were exacerbated by this particular limitation in our research setting. Future research should examine the impact those collaborative technologies in the context of geographically distributed feature-driven development.

Finally, the development locations differed in the maturity of the development practices. All the development locations but the one in India started to use a feature-oriented development approach at the same time. The Indian site, on the other hand, was constrained by their high levels of maturity in their development processes (CMM level 5). Hence, their adoption of the more fluid development practices associated with feature-oriented development was limited. However, we do not think that such disparity has a major impact in our results for two reasons. First, the amount of work done in the Indian site was about 12% of the total development effort of the project and, second, most of the work was associated with parts of a feature. Only 17 product features out of the 1195 were completely developed by teams in India.

6.2 Implications for 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, Ramasubbu 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 that 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.

6.3 Practical Implications

Our results also have important practical implications. As software architecture has become a central element in the development process of many software projects, it is quite common to have in early stages of projects, sufficient information of the software architecture, its constituent elements and their relationships. The strong impact on integration failures that cross-feature dependencies exhibited in our analyses suggest that software architects, software manager or other stakeholders are now in a position to assess the level of cross-feature interactions, determine their relative importance and plan appropriate organizational mechanisms to support the feature teams involved in developing highly interrelated features.

Second, and in relation to the previous point, the detrimental impact of technical coupling within a product feature and cross-feature dependencies is likely to be reduced, our results suggest, by adequately selecting the feature owner. Traditionally, in software development as well as in other engineering disciplines, the selection criteria for technical leadership positions tend to be based on experience and technical expertise. Certainly, our analyses do not dispute such an approach but they suggest that if such a person is closely involved with an architectural component part of a product featured that highly coupled with the rest of the components, such decision will pay off more in terms of software quality than if such person is closely involved in the development of any other component of the product feature. Then, our findings provide guidance to software managers and other decision makers of how to best select feature owners.

7. ACKNOWLEDGEMENTS

We thank the reviewers for their valuable feedback that has helped improve significantly the study reported here. The authors are also grateful for the financial support provided by Accenture, Robert Bosch and IBM that made this research possible.

8. REFERENCES

[1] Austin, R.D. 2001. The Effects of Time Pressure on Quality in Software Development: An Agency Model. *Management Science*, 12, 2 (Feb. 2001), 195-207.

[2] Basili, V.R. and Perricone, B.T. 1994. Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM*, 12 (1994), 42-52.

[3] Bird, C. et al. 2009. 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.

[4] Boh, W.F. et al. 2007. Learning from Experience in Software Development: A Multilevel Analysis. *Management Science*, 53, 8 (Aug. 2007), 1315-1331.

[5] Briand, L.C. et al. 2000. Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems. *The J. of Systems and Software*, 51 (2000), 245-273.

[6] Calder, M. et al. 2003. Feature Interaction: A Critical Review and Considered Forecast. *Computer Networks*, 41 (2003), 115-141.

[7] Cataldo, M. 2010. 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.

[8] Cataldo, M. and Herbsleb, J.D. 2009. 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.

[9] Cataldo, M. and Nambiar, S. 2009. 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.

[10] Cataldo, M. and Nambiar, S. 2010. 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.*.

[11] Cataldo, M. and Herbsleb, J.D. 2010. *Coordination Breakdowns and their Impact of Development Productivity and Software Failures.* Technical Report CMU-ISR-10-104, School of Computer Science, Carnegie Mellon Univ.

[12] Clements, P. and Northrop, L. 2002. *Software Product Lines: Practices and Patterns*. Addison-Wesley, New York, NY.

[13] Curtis, B. 1981. *Human Factors in Software Development*. Ed. by Curtis, B., IEEE Computer Society.

[14] Curtis, B., et al. 1988. A field study of software design process for large systems. *Comm. of the ACM*, 31 (1988).

[15] Espinosa, J.A. et al. 2007. Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development. *Org. Science*, 18, 4 (Aug/Sep. 2007), 613-630

[16] Fitzpatrick, G. et al. 2006. CVS Integration with Notification and Chat: Lightweight Software Team Collaboration. In *Proceedings of the Conference on Computer Supported Cooperative Work* (Banff, Canada). CSCW'06.

[17] Graves, T.L. et al. 2000. Predicting Fault Incidence Using Software Change History, *IEEE Transactions on Software Engineering*, 26 (2000), 653-661.

[18] Hahn, Y. et al. 2008. Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties. *Information Systems Research*, 19, 3 (Sept. 2008), 269-391.

[19] Harter, D.E. et al. 2000. Effects of Process maturity on Quality, Cycle Time, and Effort on Software Product Development. *Management Science*, 46, 4 (Apr. 2000), 451-466.

[20] Kiczales, G. et al. 1997. Aspect-Oriented Programming. In *Proceedings of the European Conference on Object-Oriented Programming* (Finland). ECOOP'97.

[21] King, G., L. Zeng. 2001. Logistic regression in rare events data. *Political Analysis,* 9, 2 (2001), 137-163.

[22] Krishnan, M.S., et al. 2000. An Empirical Analysis of Productivity and Quality in Software Products. *Management Science*, 46, 6 (Jun. 2000), 745-759.

[23] Kutner, M. et al. 2004. *Applied Linear Regression Models, 4th Ed*., McGraw-Hill Irwin.

[24] Louridas, P. 2006. Using Wikis in Software Development. *IEEE Software*, 23 (Mar./Apr 2006).

[25] Moeller, K.H. and Paulish, D. 1993. An Empirical Investigation of Software Fault Distribution. In *Proceedings of the International Software Metrics Symposium*.

[26] Nagappan, N. and Ball, T. 2007. Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study. In *Proc. of the 1st Int'l Symposium on Empirical Soft. Eng. and Measurement* (Madrid, Spain). *ESEM'07*.

[27] Palmer, S.R. and Felsing, J.M. 2002. *A Practical Guide to Feature-Driven Development*. Prentice-Hall, Upper Saddle River, NJ.

[28] Prehofer, C. 1997. Feature-Oriented Programming: A Fresh Look at Objects. *Lecture Notes in Computer Science*, 1241 (1997), 419-443.

[29] Ramasubbu, N. and Balan, R. K. 2007. Globally Distributed Software Development Project Performance: An Empirical Analysis. In *Proc. of the 15th Conference on Foundations of Software Engineering* (Dubrovnik, Croatia). FSE'07.

[30] Selby, R.W. and Basili, V.R. 1991. Analyzing Error-Prone System Structure. *IEEE Transactions on Software Engineering*, 17 (1991), 141-152.

[31] Trujillo, S. et al. 2007. Feature-Oriented Model Driven Development: A Case Study for Portlets. In *Proceedings of the International Conference on Software Engineering* (Minneapolis, MN). ICSE'07.

[32] Williams, C. et al. 2010. Supporting Enterprise Stakeholders in Software Projects. In *Proceedings of the 3rd International Workshop on Collaborative and Human Aspects of Software Engineering* (Cape Town, South Africa, 2010). CHASE'10.

[33] Weisband, S. 2008. *Leadership at a Distance: Research in Technologically-Supported Work*. Lawrence Erlbaum Associates, New York, NY.

[34] Wolf, T. et al. 2009. Predicting Build Failures using Social Network Analysis on Developer Communication. In *Proceedings of the International Conference on Software Engineering* (Vancouver, Canada). ICSE'09.