SOFTWARE PROCESS DIVERSITY: CONCEPTUALIZATION, MEASUREMENT, AND ANALYSIS OF IMPACT ON PROJECT PERFORMANCE

Narayan Ramasubbu, University of Pittsburgh, (narayanr@pitt.edu)
Anandhi Bharadwaj, Emory University, (a.bharadwaj@emory.edu)
Giri Kumar Tayi, SUNY, Albany, (gtayi@albany.edu)

Abstract

We investigate software process diversity, defined as the project condition arising out of the simultaneous use of multiple software development process frameworks within a single project. We analyze the antecedents and consequences of software process diversity and examine the relationship among software process diversity, organizational process compliance, and software project performance. Our conceptualization characterizes software process diversity as the response of a project team to contingencies such as requirements volatility, design and technological novelty, and the level of customer involvement in the project. We empirically tested this conceptualization utilizing a discovery-oriented approach and data collected from 410 large commercial software projects. Results show that higher levels of requirements volatility, design and technological novelty, and customer involvement increased software process diversity within a project. However, software process diversity within a project decreased with an increase in the level of process compliance enforced on the project. We also found that a higher degree of fit (or match) between process diversity and process compliance, rather than those mechanisms independently, was significantly associated with an increase in project productivity and quality. These results indicate that increasing software process diversity in response to project-level contingencies improves project performance only when there is a concomitant increase in organizational process compliance efforts. We discuss the implications of these results for research and derive prescriptive guidelines to manage the fit between process diversity and process compliance for achieving better software project performance.

Key Words: Software process diversity, process compliance, plan-based processes, agile processes, software engineering, productivity, quality, fit as matching.

Paper under review; please do not cite or distribute
INTRODUCTION

Pursuing firm-wide process standardization has traditionally been prescribed as a ‘best practice’ for software organizations to address the challenges of developing high quality software in a cost effective way (Humphrey 1989; Van der Pijl et al. 1997; Harter et al. 2000; Ramasubbu et al. 2008). Until recently, such standardization efforts have been typically implemented through a firm-wide adoption of a single plan-based or agile normative software process framework\(^1\) (Krishnan and Kellner 1999; Lycett et al. 2003; Ramasubbu 2014). Accordingly, a majority of prior research on software development processes implicitly treat project-level processes as being uniformly derived from a single normative process framework (Krishnan et al. 2000; Jiang et al. 2004; Agarwal and Chari 2007). However, project-level processes in software development can be divisible, and project teams can customize their processes by selecting and combining elements from multiple plan-based and agile normative frameworks (Fitzgerald et al. 2006; Napier et al. 2008). Practitioner reports from the field also suggest that firms are increasingly adopting multiple process frameworks for their projects and embracing software process diversity (Anderson 2005; Bella et al. 2008). Software process diversity refers to the condition of project-level processes being composed of elements drawn from multiple normative software process frameworks that could differ in their underlying attributes and philosophies (Lindvall and Rus 2000; Deck 2002).

There is growing recognition that software process diversity could aid teams in overcoming the limitations imposed by standardized processes based on a single normative process framework that advocates strict adherence to either plan-based processes or agile

---

\(^1\) Normative software process frameworks refer to a wide array of models (or methodologies) of software development published by standards organizations in the software industry such as the Software Engineering Institute, Scrum.org, and the International Organization for Standardization (ISO). These frameworks such as the Capability Maturity Model (CMM) and Scrum provide templates and guides for firms to model their software development processes. Firms often seek certifications from the industry standards bodies to signal to the market that their software processes are compliant with the prescriptions of the adopted normative frameworks.
processes (Vinekar et al. 2006; Napier et al. 2008; Magdaleno et al. 2012). Past research studies have highlighted that the combination of plan-based and agile processes within a single project, and the resulting increase in software process diversity, could help software teams to better adapt to changing user requirements and design specifications (Harris et al. 2009; Ramesh et al. 2012). Capabilities gained from embracing process diversity could also help teams address the conflicting demands in a project such as the need to be efficient versus the need to be flexible (Subramanyam et al. 2012). By increasing process diversity, software teams that traditionally use plan-based frameworks could infuse flexibility into their context by adopting certain components of agile process frameworks such as accepting requirement changes and more frequent testing (Boehm 2003; Ramasubbu and Balan 2009). Similarly, by embracing process diversity, agile software teams could incorporate some elements of structure and formal communication associated with plan-based process approaches to enhance efficiency and overall predictability of outcomes (Ramesh et al. 2012).

While examining the impacts of software process diversity on the performance of custom software development projects, it is important to recognize the presence of another countervailing force in the form of organizational process compliance enforced in those projects. Stringent compliance to established process standards is considered as an important firm capability in the custom software development industry (Ethiraj et al. 2005; Ramasubbu 2014). Software firms are known to voluntarily seek compliance verification and certification from third-party standards organizations in order to gain reputation as “high process maturity” firms (Van der Pijl et al. 1997; Gopal and Gao 2009). These firms allocate significant resources to monitor project teams for their compliance to organizational process standards (McGarry and Decker 2002; Staples and Niazi 2008; Ramasubbu et al. 2008). Such organizational process
compliance enforced on project teams can be expected to influence both the levels of process
diversity pursued by individual project teams and their subsequent performance. However, there
is a dearth of research examining the impacts of software process diversity on the performance of
projects executed in an environment of stringent process compliance mandates. We address this
gap through this study.

We raise the following research questions: What factors contribute to an increase in
software process diversity in real world projects that are also actively monitored for compliance
to mandated process standards? What is the joint effect of software process diversity and process
compliance on project performance? To answer these research questions, we proceeded as
follows. We theoretically conceptualized the software process diversity construct and developed
its empirical formulization. Then, we adopted a discovery-oriented research approach to
facilitate model development by integrating theory-based perspectives and field-based insights
elicited from industry practitioners (Menon et al. 1999; Tuli et al. 2007). We partnered with a
leading multinational software firm for in-depth field observations of software projects. By
iterating insights from our field observations and in-depth interviews of software team members
with the corresponding research literature, we developed a conceptual model of the antecedents
and performance implications of software process diversity. Subsequently, we tested and verified
our conceptualization utilizing archival data from 410 software projects completed at our
research site. Finally, based on the results from the analysis and from follow up discussions with
project managers from the field, we developed a set of prescriptive guidelines for balancing
process diversity and process compliance in software projects.

A CONCEPTUALIZATION OF SOFTWARE PROCESS DIVERSITY

When software teams adopt and fuse multiple normative process frameworks within a
single project, they increase the overall software process diversity within the project. While there
have been anecdotal and case descriptions of software process diversity in prior research (e.g., Boehm 2003; Jakobsen and Jhonson 2008; Jakobsen and Sutherland 2009), a rigorous analysis of the concept of software process diversity, its relationship with organizational process compliance, and their joint impact on project performance is absent in the literature.

To address this gap, we draw on the broader management literature which has focused on various forms of organizational diversity such as demographic differences, differences in values, functional skills, and personality types. Harrison and Klein (2007) review the literature on organizational diversity and provide guidelines for its conceptualization and measurement. They recommend conceptualizing diversity as the differences in the beliefs and values of team members (diversity as separation), the differences in the knowledge, experience and perspectives of team members (diversity as variety), and the differences in the influences and resources held by team members (diversity as disparity). Building on this, we theoretically define software process diversity as the composition of differences in separation, variety, and disparity of key process areas implemented in a software project.

A key process area (KPA) is a cluster of related activities (or tasks) in a project that, when performed collectively, achieves a set of goals for successful software development (Ramasubbu et al. 2008). Normative software process frameworks prescribed by industry-based standards organizations clearly specify the number and scope of KPAs that they support. For example, the CMMI framework for software development consists of a total of 22 KPAs covering the entire lifespan of a development project\(^2\). When a software organization adopts a normative process framework such as the CMMI, a process template covering the KPAs of the framework is instituted. This process template provides a predefined collection of artifacts that

can be used to organize the workflow spanning the KPAs throughout a project’s lifecycle. In organizations that allow project teams to simultaneously use multiple normative frameworks, process templates corresponding to the different frameworks are made available to a software project team. Software teams typically activate specific components of each of those templates for use in their projects (Fitzgerald et al. 2006; Ramasubbu et al. 2008). Thus, in this scenario, multiple process templates collectively implement the entire set of KPAs of a project. Since the use of process templates by a software team is logged and traceable, longitudinal observations of the membership of a project’s KPAs to the different process templates provides an empirical basis to reliably measure software process diversity and the underlying dimensions of separation, variety, and disparity of the KPAs implemented in a project.

The separation dimension of software process diversity measures the composition of differences among KPAs in a software project belonging to process templates derived from plan-based versus agile normative process framework. This is analogous to assessing diversity in a project team by acknowledging the differences in knowledge bases, perspectives, and deeply held values and beliefs among the team members (Kilduff et al. 1995; Homan et al. 2007). Plan-based and agile process frameworks vary in their overall values, beliefs, and practice attitudes (Boehm 2002; Boehm and Turner 2003). For example, while plan-based frameworks such as the CMMI favor structured planning processes and comprehensive documentation for traceability, agile frameworks such as Extreme Programming tend to emphasize “individuals and interactions over tools, working software over documentation, customer collaboration over contract negotiation, and responding to change over following a plan” (Beck et al. 2001). Measuring the composition of KPAs belonging to the plan-based and agile process templates in a project captures the extent to which a project’s KPAs are diverse in their process values and beliefs.
The *variety* dimension of software process diversity measures the composition of differences in the *spread* of KPAs across the different process templates used in a project. This accounts for differences among the varieties of process frameworks used in a project even when they are not separated in their overall philosophy (e.g., plan *versus* agile). For example, a software project might utilize templates from two plan-based frameworks such as the CMMI and Rational Unified Process (RUP), and allocate to each 50% of the total project KPAs. Another project in the firm might utilize the same two process templates, but allocate 25% of its KPAs to the CMMI process template and 75% of its KPAs to the RUP template. The separation diversity score for these two projects would be zero as both CMMI and RUP impose a plan-based development paradigm and there is no separation of values in this case. However, the two projects still differ in the extent to which they allocate KPAs to the two varieties of process templates (CMMI and RUP). Thus, there is a need to capture differences among software projects with respect to the number of process templates they use and the way they allocate KPAs to those different varieties of process templates. The variety dimension of software process diversity accomplishes this by capturing the spread of the KPAs among the different varieties of process templates used in a project.

The *disparity* dimension of software process diversity measures the composition of differences in the *resource allocation* to the KPAs belonging to the different varieties of process templates utilized in a project. Measuring disparity in resource allocation is important because it captures the extent to which the diverse elements in a process were actually utilized in project activities. Resource allocation in software projects has been typically measured as the percentage of total project effort allocated to a particular activity (Krishnan *et al.* 2000; Ramasubbu *et al.* 2008). Accordingly, the disparity dimension of process diversity accounts for resource allocation...
differences by considering the differences in the project effort allocated to the different KPAs and the process templates they are associated with.

In summary, rather than assuming that the KPAs of a project are homogenously distributed with respect to the underlying normative process framework(s), the construct of software process diversity allows us to account for the separation, variety, and disparity of software processes within a project. Next, we develop a conceptual model of the antecedents and consequences of software process diversity and formulate our key hypotheses.

**RELATING SOFTWARE PROCESS DIVERSITY, PROCESS COMPLIANCE, AND PROJECT PERFORMANCE**

*Discovery-Oriented Research Approach*

To develop the conceptual model of this study we followed a discovery-oriented research approach that facilitates the uncovering of theories-in-use or forms of knowledge embedded in the realm of practitioners (Menon *et al.* 1999; Tuli *et al.* 2007). We first formulated the basic tenets of a preliminary model based on the research findings reported in the information systems development and software engineering literatures, and then refined it based on in-depth discussions with practitioners in the field.

Keeping in mind the dual forces of software process diversity and process compliance in high process maturity environments, we began our model formulation by building on prior research findings on “controlled-flexible” process designs (Harris *et al.* 2009). Controlled-flexible process designs aim for both process improvisation and establishment of disciplined management controls to enforce compliance. Using case studies, Harris *et al.* (2009) established evidence for the superior performance of controlled-flexible process designs when compared to pure plan-based designs. Those case studies revealed cross-project variations in the way controlled-flexible process designs were enacted by different project teams under the influence
of technological and market uncertainties. Addressing how teams could adapt to such uncertainties, Vidgen and Wang (2009) studied software process design through the lens of complex adaptive systems and discussed the coevolution of software processes along with the business environment. In the software engineering literature, project-level contingencies such as requirements volatility, degree of customer involvement, and code size have been reported as influencing specific process design choices made by a software team (Boehm and Turner 2003; Ramasubbu and Balan 2009; Magdaleno et al. 2012). Finally, prior research has also established that high process maturity organizations, such as those assessed at CMM level-5 process maturity, enforce stringent process compliance mechanisms to ensure minimal deviance from prescribed standards (Ethiraj et al. 2005; Ramasubbu et al. 2008). Process compliance efforts tend to reduce process variations within a project, and they have been associated with the benefits of minimizing schedule deviation and lowering the number of defects in the delivered software (Krishnan and Kellner 1999; Harter et al. 2000; Harter et al. 2012).

The above research findings motivated our conceptual model development as follows: (1) depending on specific project-level contingencies, software teams make choices regarding the underlying normative process frameworks for their projects, which could result in increased levels of software process diversity within the projects; and (2) process diversity and process compliance efforts expended on a project jointly impact project performance outcomes.

To enrich our preliminary conceptualization, we conducted in-depth discussions with practitioners in the field. We collaborated with a leading multi-national software development firm that was a recipient of the IEEE Software Process Achievement Award (IEEE SPA 2010) and had an industry-wide reputation as a leader in process innovation. The firm operated in 55 countries with over a hundred thousand employees and about 8 billion dollars in annual revenues.
at the time of our data collection. All the development centers from which we collected data for this study were assessed as operating at CMM level-5 process maturity by an independent auditing agency. As is common in a discovery-oriented research method, we took our initial conceptualization to the field and engaged with practitioners through focus group meetings and structured interviews. Our goal was to perform a qualitative face validity test of the initial model and to discover underlying mechanisms that influence the relationship between software process diversity, process compliance, and project performance.

Interacting with a total of 109 practitioners at our research site, we conducted 27 focus group meetings and 28 one-on-one, structured interviews. Our discussions and interviews involved participants describing their team’s development environment, process design rationale, and lessons learned from their experience. Through these field interactions, we discovered temporal variations in the way different project teams situated themselves in the plan based-to-agile process design continuum during the lifespan of their projects. While prior case studies in the literature had documented cross-project variations in process design, we were able to confirm the existence of significant within-project variation in process designs. Discussions with practitioners also highlighted the influential role played by centralized and independent organizational units such as the software engineering process group (SEPG). At our research site, the SEPG was empowered to act as an autonomous body that conducted rigorous internal audits on project teams to enforce compliance to the firm’s process standards. These discussions informed the further development of the study’s hypotheses described below.

**Antecedents and Consequences of Process Diversity**

As noted earlier, software process variations within a project reflect the choices made by the software team in response to specific contingencies encountered during the project life cycle.
Prior research on software project risk has examined how specific project-level contingencies (or project risk factors) contribute to overall project risk and impact performance outcomes (Barki et al. 1993; Keil et al. 2000). Much of this work characterizes project risks as stemming from various types of uncertainties related to project requirements, end user involvement, underlying technological and design complexity, project team composition, and the overall organizational environment (Wallace et al. 2004). We draw on the risk framework proposed by Wallace et al. (2004) to develop specific hypotheses relating project risk factors and software process diversity. In doing so, we characterize the extent of process diversity observed in a project as the collective response of the project team to the perceived project risks through a key mechanism they control, namely, process design. However, as noted earlier, a countervailing choice exercised by SEPG in the form of process compliance mechanisms attempts to reign in the within-project process diversity. Software process diversity is therefore the result of both project-specific risk factors as well as the extent of organizational process compliance enforced on the project.

In contrast to prior studies that have examined project risks in relation to overall project performance, we focus on the intermediate outcome of software process diversity, which in our conceptualization is a key conduit through which project risk factors impact project performance. Accordingly, among the six dimensions of project risk identified by Wallace et al. (2004)³, we develop hypotheses pertaining to requirements risk, design and technology risk, and end-user (customer) involvement risk. These factors were emphasized as the key triggers for increasing software process diversity by practitioners at our research site during our discussions. We account for the remaining risk factors through appropriate control variables, such as software project size, team size, and personnel experience, in our empirical models.

³ The six dimensions are organizational environment risk, user risk, requirements risk, project complexity risk, planning and control risk, and team risk.
The final component of our conceptualization is the joint impact of process diversity and process compliance on project performance. In hypothesizing this effect, we consider two project performance dimensions widely examined in the information systems development literature: productivity and quality (Krishnan et al. 2000; Harter et al. 2000; Ramasubbu et al. 2008). We utilize the “fit as matching” perspective (Venkatraman 1989) to characterize the fit (or match) between process diversity and process compliance that impacts project performance. Conceptually, project performance improves with a good fit. That is, when a higher (lower) level of process diversity is matched with a higher (lower) level of investment in process compliance, there is an improvement in productivity and quality. Table 1 presents a summary of the key hypothesized constructs and Figure 1 shows our overall conceptual model.

**Table 1. Key Constructs in Conceptualization**

<table>
<thead>
<tr>
<th>Model Component</th>
<th>Construct</th>
<th>Description</th>
<th>Correspondence to Wallace et al. (2004) risk dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antecedents of software process diversity</td>
<td>Requirements volatility</td>
<td>Uncertainty surrounding project requirements and frequent change in requirements</td>
<td>Requirements risk</td>
</tr>
<tr>
<td></td>
<td>Design and technological novelty</td>
<td>The level of newness and unfamiliarity with the system design and implementation technology</td>
<td>Project complexity risk</td>
</tr>
<tr>
<td></td>
<td>Customer involvement</td>
<td>The level of involvement from key stakeholders of the client firm</td>
<td>User risk</td>
</tr>
<tr>
<td></td>
<td>Organizational process compliance</td>
<td>The extent of effort spent on meeting process standards mandated by the firm</td>
<td>Planning and control risk</td>
</tr>
<tr>
<td>Performance implications of software process diversity</td>
<td>Fit (misfit) between process diversity and process compliance</td>
<td>The degree of congruence (deviance) between the extent of process diversity and efforts spent on process compliance activities in a project</td>
<td>--N/A--</td>
</tr>
<tr>
<td></td>
<td>Productivity</td>
<td>Overall efficiency in delivering the project (output/input)</td>
<td>Project performance</td>
</tr>
<tr>
<td></td>
<td>Quality</td>
<td>Overall quality of delivered software (1/defects delivered)</td>
<td>Project performance</td>
</tr>
</tbody>
</table>

**Requirements Volatility**

Requirements volatility refers to the extent to which the functional and non-functional requirements for a software system being developed change during the project’s lifespan (Barry et al. 2006; Agarwal and Chari 2007). Requirement changes and subsequent alterations of
software code have been reported to create “ripple effects” of rework in a project (Harter et al. 2000; Mookerjee and Chiang 2002). Rework might result due to legitimate changes in a client’s needs, but can also stem from ambiguous, inadequate or wrong articulation of requirements in the first place (Wallace et al. 2004).

**Figure 1. Conceptual Model**

Plan-based and agile processes vary in their approaches to handle requirements volatility, the underlying causes, and subsequent ripple effects of rework. Plan-based approaches primarily work towards minimizing requirements volatility through a combination of formal contracts that evoke client commitments and partitioning mechanisms that “freeze” requirements over certain stages of a project’s lifecycle. In contrast, agile processes tend to respond to requirements volatility through improvisation, rapid adaptation, and incremental delivery of software (Boehm 2002). The rapid adaptation and social approach inherent in agile process framework, however, have been noted as not scalable due to inefficiencies of frequent informal meetings (Begel and Nagappan 2007; Dyba and Dingsoyr 2008). Thus, in the presence of requirements volatility, there are tradeoffs to be made when any single framework is used exclusively in a project.

One way to remove the constraints of such tradeoffs, then, is to adopt both plan-based and agile process frameworks within a project. Using multiple process frameworks would help
project teams to better handle the fluidity in requirements as they could cordon off parts of the project that have different levels of volatility and tackle them through different process approaches. For example, teams could partition a project into different phases (or iterations) of high and low volatility to use different process frameworks for these phases (MacCormack and Verganti 2003). Teams could also partition the software into different modules according to the requirements volatility levels and suitably apply different varieties of processes to them (MacCormack et al. 2001).

Use of different varieties of process templates from both plan-based and agile frameworks creates separation in the overall philosophies towards software development within a project. Also, it has been observed that requirements volatility may not occur homogeneously throughout a project’s lifecycle and it does not affect all parts of a system’s development in a uniform way (Banker and Slaughter 2000; Barry et al. 2006). The heterogeneous patterns of requirements volatility, in turn, cause disparity in effort expenditure on the different project phases, partitioned systems parts developed during those phases, and their associated process templates. Thus, an increase in the requirements volatility of a project tends to increase the variety, separation, and disparity of processes within the project, which leads to an overall increase in software process diversity. Therefore,

\[ H1. \text{A higher level of requirements volatility in a project is associated with a higher level of software process diversity.} \]

Design and Technology Novelty

Project teams facing new software design and technologies face a steep learning curve during the development process (Kemerer 1992; Boh et al. 2007). Initially, such teams might employ agile processes with rapid iterations to gain early feedback and accelerate team learning (MacCormack et al. 2001). However, over time as the new technological features become more
familiar and tasks get predictable, the focus can shift to a plan-based approach that facilitates better efficiency and benefits due to scale economies of longer iterations (Banker et al. 1997). Furthermore, the use of novel design and technologies in a software project could be a significant contributor to project complexity resulting in higher levels of project management risk (Wallace et al. 2004). Normative process frameworks that imbibe an exclusive plan-based or agile paradigm advocate different approaches toward mitigating such risks. While plan-based frameworks often prescribe explicit risk management processes, risk management in agile process frameworks often remains a passive activity (Boehm 2003; Moran 2014).

Higher levels of design and technological novelty present a demanding requirement on software teams to both accommodate quick iterations to facilitate effective learning and implement a planned approach to manage risks. Teams facing such challenging requirements resort to combining the attributes of multiple varieties of process frameworks as it helps them to develop ambidextrous capabilities (Napier et al. 2008; Magdaleno et al. 2012; Ramesh et al. 2012). In such a scenario, the variety of process frameworks chosen by a project team would include a number of agile-oriented process components because rapid iterations facilitated by agile processes aid efficient learning and effective client feedback during early project stages (MacCormack et al. 2001). Inclusion of agile-oriented process component in high process maturity environments, where the use of structured and plan-based approaches is the organizational norm, naturally leads to a separation of values and beliefs within a project.

Furthermore, learning curve effects resulting from higher levels of design and technological novelty cause variance in the effort expended on project tasks over the lifecycle of the project (Kemerer 1992). As teams learn and get familiar with the new designs and technologies, effort required to complete project tasks tends to decrease, although the patterns of
such a decrease may not be uniform during the different project stages (Boh et al. 2007). This variance in effort expenditure corresponds to a variance associated with the use of different varieties of process templates during the early and late project stages. That is, there is disparity in the allocation of resources to the different varieties of process templates utilized during the various project lifecycle stages. Thus, an increase in design and technological novelty increases the variety, separation, and disparity of processes within a project and therefore,

\[ H2. \text{ A higher level of design and technological novelty in a project is associated with a higher level of software process diversity.} \]

**Customer Involvement**

The level of customer (end-user) involvement in a project and how this influences the project’s processes and performance repeatedly came up in our discussions with team members at the research site. Prior research findings also lend strong support to a link between end-user involvement and project-level processes and outcomes. Wallace et al. (2004) categorize the lack of user involvement in a project as a significant project risk and note that such risks could contribute to project failures. Exploring customer-developer links in software projects, Keil and Carmel (1995) show that projects with a higher degree of participation from customers tend to perform better. Such project success has been attributed to the ability of software teams to match their processes to customer needs through appropriate changes to development processes (MacCormack and Verganti 2003). On the one hand, when there is a high degree of customer participation in a project, team members might be expected to more actively respond to evolving customer requirements (Subramanyam et al. 2010). On the other hand, a higher degree of customer participation might also necessitate the need for more formal communication plans, especially when there are organizational boundaries between the customer and the vendor teams (Pikkarainen et al. 2008; Ramesh et al. 2012).
To respond to such dual needs in managing customer involvement in a project, software teams could draw appropriate process components from multiple normative frameworks for their projects. As highlighted in prior research, a higher degree of customer involvement positively influences a project team’s orientation towards selecting more agile process components (Boehm and Turner 2003; Ramasubbu and Balan 2009). Naturally, in the context of custom software development executed by high process maturity firms, this leads to a situation where a software team has to grapple with separate value systems enshrined in the different varieties of process templates used in the project. Furthermore, the degree of client involvement in a project may not be uniform throughout the different phases of a software project (Subramanyam et al. 2010). There are also variations in the expectations of different end-users of a customer firm with respect to the formalism required in project-level communications and the corresponding coordination mechanisms (Nidumolu 1995; Andres and Zmud 2002). These variations result in overall heterogeneity in effort expenditures across formal (documents, contracts, etc.) and informal (mutual adjustments, informal interaction, etc.) coordination schemes. In turn, these would cause a corresponding disparity in the effort allocation to the different KPAs belonging to the plan-based and agile process templates that respectively facilitate formal and informal coordination. Thus, we expect the degree of customer involvement to positively influence the variety, separation, and disparity of processes of a project team. Therefore,

**H3. A higher level of customer involvement in a software project is associated with a higher level of software process diversity.**

**Process Compliance**

Process compliance mechanisms are procedures enacted to ensure the fidelity of the software design processes to the selected process frameworks (such as adherence of the team to the normative prescriptions of the CMMI framework). Process compliance is typically enforced
by quality control groups, charged with the responsibility of auditing and evaluating adherence to quality metrics including the normative prescriptions for process design and documentation (Ogasawara et al. 2006; Ramasubbu 2014). In high process maturity environments, project teams are expected to report all process tailoring activities, and they are monitored and evaluated through internal audits and related quality assurance activities. Thus, process compliance efforts contribute to an indirect selection mechanism wherein a deliberate investment is made to monitor the project-level processes and assess their fit with the overall project and firm context. They act as a stabilizing force that tends to limit process diversity by blocking process variations that are deemed unfit, unnecessary, or non-compliant with firm standards, as evaluated by the independent quality assurance personnel.

Apart from pruning project-level process variations, higher levels of process compliance enforcement can also provide learning opportunities for reconciling disparate values among project members (Ramasubbu et al. 2008). This can result in project teams utilizing more cohesive process components that are not too distinct in their underlying value attributes. Also, with a higher level of process compliance, process components that do not receive significant levels of resource commitments from the project team could be identified as potentially less useful variations that can be eliminated without significantly impacting project outcomes. Thus, we expect process compliance to act as a countervailing force to software process diversity by reducing the variety, separation, and disparity of processes in a software project. Hence,

\[ H4: \text{A higher level of process compliance effort in a software project is associated with a lower level of software process diversity.} \]

**Fit Between Process Diversity and Process Compliance**

The discussion up to this point has focused on examining the project-level contingencies that increases software process diversity and the role of process compliance in reining in process
diversity. In this section, we examine the fit (or match) between process diversity and process compliance and examine their joint effects on overall project performance. Following well-established protocols in prior literature for measuring software project performance (e.g., Krishnan et al. 2000; Harter et al. 2000; Ramasubbu et al. 2008), we conceptualize project performance along the two dimensions of development productivity of a project team and the conformance quality in the software code delivered by the project team.

Development productivity refers to the efficiency of a project team in developing and delivering software code using resources available to the team (Krishnan et al. 2000). When lower levels of process diversity in a project are matched with a high degree of process compliance investments, it results in suboptimal use of organizational resources. The excessive compliance efforts in the above scenario could have otherwise been used for core project tasks such as software development and verification. Similarly, when a project team is subjected to lower level of investments in organizational process compliance, but operates using a higher level of process diversity, there is a mismatch and a negative impact on productivity. This is because insufficient process compliance efforts often lead to the propagation of suboptimal or bad practices that cause rework during the late stages of a project (Stamelos 2010). The costs of such rework are difficult to contain because of complex dependencies that cause ripple effects through all parts of the software system that is being developed (Mookerjee and Chiang 2002). Such additional expenditures due to rework ripples dent the overall development productivity of a project. Similarly, when higher levels of process diversity are met with lower levels of process compliance efforts, there are insufficient resources to detect software errors committed during development (i.e., lower levels of conformance quality). Such undetected development errors are likely to be reported by customers after the delivery of software code, for example, during
acceptance testing. Rectifying software errors after delivery is challenging and leads to an overall increase in project rework and cost of quality (Harter et al. 2000).

In contrast, a good fit between process diversity and process compliance enables a project team to optimally balance the increased flexibility resulting from the use of multiple process frameworks and the rigor needed to detect and rectify software development errors before delivery. Such an optimal balance has been reported to engender process-based learning in software development that helps teams to realize improvements in productivity and reduce the number of defects delivered to customers (Ramasubbu et al. 2008; Subramanyam et al. 2012). A good fit between process diversity and process compliance also leads to the development of co-evolutionary change mechanisms that help project teams adapt to uncertain business situations in an efficient way (Volberda and Lewin 2003; Vidgen and Wang 2009). Thus, a better fit between process diversity and process compliance endows project teams with capabilities that help them cope with project contingencies in an efficient way and reduce defects (or improve quality) through improved learning and effective quality control mechanisms. Therefore, our final set of hypothesis is:

\( H5a: \) A higher degree of fit between process diversity and process compliance in a software project is associated with a higher level of project productivity.

\( H5b: \) A higher degree of fit between process diversity and process compliance in a software project is associated with a higher level of software quality.

**ANALYSIS AND RESULTS**

**Data and Variables**

We collected data from 410 commercial software projects completed by a multinational software vendor firm in a recent two year period. The data had been verified multiple times by the firm’s external auditors for quality certification purposes (CMMI level-5 compliance and IEEE SPI award). Thus, we were able to utilize a unique and highly reliable data for our analysis.
**Process Diversity:** As noted earlier, we consider a KPA as the fundamental unit of a software process design. At our research site, a majority of the software development centers used the 22 KPAs prescribed for CMMI level-5\(^4\) development operations. Other normative process frameworks that project teams subscribed to during the time period of this study were IBM’s Rational Unified Process (RUP), Agile RUP, Scrum, and Extreme Programming (XP). The baseline process templates for all of these five normative process frameworks were standardized and controlled by the firm’s centralized SEPG. Individual projects were allowed to implement all or a subset of the five process templates for their specific use. Therefore, software teams had the flexibility to both choose a set of relevant process templates for their project and design the spread of KPAs across the process templates, which resulted in project-specific customizations of plan-based and agile frameworks. To be able to monitor and control such customized processes through uniform metrics, the firm’s SEPG teams mapped the KPAs of the normative process frameworks under a seamless and uniform process compliance framework\(^5\). The project-level process customizations and the spread of KPAs to the different process templates formed the basis for the measurement of the process diversity construct. For each of the KPAs implemented in a project, we observed the membership of the KPA to the corresponding process template and traced its use throughout the project’s lifecycle. Then, we calculated the separation, variety, and disparity process diversity scores for each project.

The separation dimension was derived using a standard deviation-based measurement scheme. Assigning an agile score for each KPA of a project (1 if the KPA belongs to an agile process template, 0 otherwise), the separation dimension of process diversity is derived as:

\[^4\]CMMI is an integrated process improvement framework developed by the Software Engineering Institute at the Carnegie Mellon University. Level-5 of the CMMI is the highest maturity level indicating quantitatively controlled and well-optimized processes.

\[^5\]A generalized example of such a mapping of CMM KPAs and Extreme Programming framework can be found in Paulk (2001).
\[ Separation = \sqrt{\frac{\sum_{i=1}^{s} (agileScore_i - \text{agileScore}_{\text{mean}})^2}{s}} \]

where, a project uses \( s \) KPAs belonging to process templates derived from either a plan-based normative framework (in which case the agile score=0) or an agile-based process framework (agile score=1). For example, consider a scenario where three different projects use 22 KPAs each for software development; project-1 uses only CMMI; project-2 implements 12 KPAs using CMMI and 10 KPAs using RUP; and project-3 uses CMMI for 12 KPAs and SCRUM for 10 KPAs. The separation diversity score for both project-1 and project-2 would be zero as there is no separation among the KPAs in these projects (they all belong to plan-based process frameworks, either CMMI or RUP). In contrast, for project-3 the separation score is 0.498, indicating the presence of a plan-based (CMMI) and agile (SCRUM) process separation among the KPAs of the project.

The variety dimension of diversity has been traditionally measured using the Blau’s index (Harrison and Klein 2007; Gibbs and Martin 1962), which we adapted to the context of software process diversity as follows:

\[ Variety = 1 - \sum_{i=0}^{v} p_i^2 \]

where, \( p_i \) would be the proportion of the \( s \) KPAs that belong to the \( v \) variety of process templates. Consider the previous example scenario where three different projects use 22 KPAs each for software development; project-1 uses only CMMI; project-2 implements 12 KPAs using CMMI and 10 KPAs using RUP; and project-3 uses CMMI for 12 KPAs and SCRUM for 10 KPAs. As explained above, the separation diversity scores for projects-1, 2, and 3 would be 0, 0, and 0.498 respectively, which captures the separation of KPAs into plan-based and agile

---

6 The mean agile score for project-3 is 0.45 and the overall separation score is the standard deviation of the agile scores (10 counts of 1 and 12 counts of 0, yielding a standard deviation of 0.498).
categories. On the other hand, the variety diversity scores for the projects 1, 2, and 3 would be 0, 0.496, and 0.496 respectively, which captures the different patterns of spread of KPAs across different process templates, but unlike the separation diversity score, it does not explicitly capture the separation of plan-based KPAs and agile KPAs.

Disparity dimension of diversity has been typically measured using a coefficient of variation (Harrison and Klein 2007), which we adapted to the context of this study as follows:

\[
\text{Disparity} = \sqrt{\frac{\sum_{v=1}^{v} \left( \frac{\text{effort}_i - \text{effort}_{\text{mean}}}{\text{effort}_{\text{mean}}} \right)^2}{\text{effort}_{\text{mean}}}}
\]

where, \(\text{effort}_i\) is the percentage of total project effort spent on KPAs belonging to a \(v\) variety of process templates. Expanding the previous example scenario where three different projects used 22 KPAs each for software development, let us say project-1 uses only CMMI and allocates 100\% of its effort to it; project-2 implements 12 KPAs using CMMI and 10 KPAs using RUP, and allocates 65\% of project effort to CMMI KPAs and the rest 35\% to RUP KPAs; and project-3 uses CMMI for 12 KPAs and SCRUM for 10 KPAs, and allocates 70\% of project effort to the CMMI KPAs and the rest 30\% to the RUP KPAs. As discussed before, the separation diversity scores for the projects 1-3 are 0, 0, and 0.498 respectively; the corresponding variety diversity scores are 0, 0.496, and 0.496. The disparity scores for the projects 1-3 are 0, 0.424, and 0.566 respectively, which capture the differences in the effort allocation between the different varieties of process frameworks used in the projects.

**Process Compliance:** Process compliance efforts in a project involve the monitoring, assessment, and auditing of process engineering activities in a project. The process compliance personnel at our research site reported to both the project manager of a software team as well as

---

7 Project-1 variety score = (1-1)/2=0; project-2 variety score = 1-(12/22)^2-(10/22)^2=0.496; project-3 variety score=1-(12/22)^2-(10/22)^2=0.496.

8 Project-1 disparity score = √((100-100)^2)/100=0; project-2 disparity score = √(((65-50)^2+(35-50)^2)/2)/50=0.424; project-3 disparity score=√(((70-50)^2+(30-50)^2)/2)/50=0.566.
to the compliance supervisor at the SEPG. Although compliance personnel had sufficient autonomy within a software project team, executing process compliance activities required them to collaborate with individual developers. The extent of resources allocated to process compliance varied across the projects and we observed a corresponding variance in the extent to which the process compliance personnel were able to systematically monitor and assess process variants across projects. We had access to the task-level accounting and work log of all SEPG personnel participating in a software project, and we were able to identify and trace the overall process compliance effort expended in a project. Similar to prior research where investments in quality management processes where measured as a percentage of total project effort (Ramasubbu et al. 2008), we derived the process compliance variable as the percentage of total project effort spent on process compliance activities.

**Fit between Process Diversity and Process Compliance:** To derive the fit variable, we follow the “fit as matching” perspective developed in prior literature (cf. Venkatraman 1989; Tang and Rai 2014). Fit as matching technique enables us to derive the match between process diversity and process compliance without reference to an external criterion variable (such as project performance). The criterion-independent matching perspective is particularly apt for our study because process diversity and process compliance efforts are primarily influenced by different and independent actors in a project who have different goals and incentives. While process diversity is driven by project-level contingencies faced by software teams, process compliance efforts are driven by standards mandated at the organizational level that are enforced by central SEPG agents.

We utilized the deviation score and residual score approaches to measure the match between process diversity and process compliance (cf. Venkatraman 1989). In the deviation
score approach, misfit (or mismatch) is derived as the absolute difference between the standardized scores of process diversity and process compliance variables. Inversing the sign of the misfit variable obtained from the deviation score approach reflects fit, and is used for easier interpretation of the regression coefficients. In the residual score approach, we derive the fit variable using the residuals from the regression of process compliance on process diversity. To verify the robustness of results, we also used the residuals from the regression of process diversity on process compliance to derive the fit variable as recommended in the literature (cf. Dewar and Werbel 1979).

**Project performance:** The dependent construct in our model, project performance, is measured using two variables, namely, software productivity and quality. To derive the productivity variable, we first measured project size using function points, which is a programming language-independent measure of software functionality (Kemerer 1993). Then, we calculated productivity as an efficiency measure by treating the total function points produced by a software team as the output of a project and total project effort as the input. Similar to prior research, we utilized the log transformed productivity variable for the regression estimations (Harter et al. 2000; Krishnan et al. 2000; Ramasubbu et al. 2008). We derived the second performance measure, software quality, as the inverse of the total number of unique software defects reported by customers (Subramanyam and Krishnan 2003).

**Requirements Volatility:** This variable measures the extent to which requirements changes from customers induced rework in a software project. We calculated requirements volatility as the percentage of effort spent on rework due to change in customer requirements, holding the planned development effort before the change occurred as the baseline.

**Design and Technology Novelty:** The extent to which the software design and
technology involved in a project was new to the project team is self-reported by project managers and programmers using a questionnaire adapted from Takeishi (2002). We sought at least two survey responses from each project team, one from a manager and the other from a technical team member. The survey asked the participants to rate the extent to which their project involved “design newness” and familiarity with the technology used to implement the designs. The responses were measured using a 0-100 point scale with ranges from old design (0%), some modifications to old design (less than 30%), to radically different than prior designs (100%). Similarly, technological familiarity was measured using scales that ranged from completely familiar to old technologies used (0%), to completely new and unfamiliar technology (100%). The responses to the design newness and technology familiarity items were highly correlated (0.94), and we took an average of those item scores to calculate the value of design and technology novelty in a project.

**Customer Involvement:** We calculated the extent of customer involvement in a project as the percentage of total project effort spent by a software team in engaging with end-users. We systematically parsed the daily time logs submitted by software project members that were utilized for billing and cost accounting purposes of the firm to identify the time allocation of individual team members for client-facing activities. Then, we aggregated the total time spent on these client-facing activities at the project level. We also identified and accounted for the presence of clients in team meetings, peer reviews, and quality inspection meetings that were not specifically reported as client-facing activities in the time logs by matching the meeting participant logs with the list of project personnel we obtained from project managers.

**Control Variables:** As mentioned before, we used software code size measured using Function Points to derive the productivity variable. Similar to prior research, we used log-
transformed code size in the regression analysis as a control variable (Harter et al. 2000; Krishnan et al. 2000). We utilize the full-time equivalent count of personnel involved in a project as the team size control variable to account for the extent of coordination and administration needs of the project. Since the cumulative experience of a project team could potentially influence the overall project performance (Boh et al. 2007), we included the average professional work experience of the project team (in years) in the regression models. Finally, some project managers at our research site had obtained professional certifications on specific normative process frameworks (e.g., certified Scrum Master). To account for this difference between the managers, we included a categorical control variable in the models, coded as 1 if a project manager possessed any professional certifications and as 0 otherwise.

**Estimation and Results**

The functional forms of the various relationships embedded in the conceptual model (Figure 1) are shown in equations 1-3. Equation 1 shows that process diversity within a project is a function of project-specific conditions and process compliance. Equations 2 and 3 indicate that the project performance outcomes, productivity and quality, are a function of fit between process diversity and process compliance after controlling for other project variables. To establish the robustness and validity of the results using the fit as matching perspective, original process diversity and process compliance variables are also included in equations 2 and 3 apart from the match variable derived from these individual components. We also included the absolute level of agile process KPAs used in a project in equations 2 and 3 to control for an alternate explanation that process agility might drive performance\(^9\).

\(^9\) We thank an anonymous reviewer for this suggestion. We measured the agile KPA score control variable as the percentage of total KPAs of project implemented using agile process frameworks.
Before estimating the regression equations, we examined the empirical relationship between the three process diversity scores, namely, separation, variety, and disparity scores. Pairwise correlation between the three scores (shown in Table 2) revealed a high degree of correlation (at p<0.01) between them.

<table>
<thead>
<tr>
<th>Diversity Score</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separation</td>
<td>1</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>Variety</td>
<td>2</td>
<td>0.79</td>
<td>1.00</td>
</tr>
<tr>
<td>Disparity</td>
<td>3</td>
<td>0.78</td>
<td>0.80</td>
</tr>
</tbody>
</table>

We examined the covariance patterns between the three diversity scores in our sample and used a series of sphericity tests and confirmed that their covariance patterns were similar. The high correlation and similar covariance structures between the three diversity scores precluded us from differentiating the distinct effects of separation, variety, and disparity dimensions in our analysis. One reason for such a high correlation between the different diversity scores could be because all the projects at our research site during our observation period existed
in a CMMI-level 5 organizational environment and moved towards higher levels of process diversity from that baseline. So project-level variations that caused process variety also simultaneously caused a separation of KPAs into plan-based and agile frameworks in a project. Further, the effort allocation across the different varieties of process templates followed the same pattern as the proportion of KPAs allocated to the different varieties of process templates in the project. Therefore, we are not able to sufficiently tease out the differences between the separation, variety, and disparity dimensions of software process diversity using our empirical data. As a result, we only infer our results using each of the separation, variety, and disparity scores individually and mutually exclusively in the regressions. Results were similar across the models where the separation, variety, and disparity scores were separately used as the process diversity variable. Also, the process diversity variable derived as a linear combination (additive, average, etc.) of the separation, variety, and disparity scores yielded similar results\(^\text{10}\).

The summary statistics and correlations between the variables utilized in the regression models are presented in Tables 3 and 4 respectively. We note that the fit variables derived using the deviation score and the residual score approaches are highly correlated with each other (0.83) as expected. However, the fit variables have relatively lower levels of correlation with the individual components that are used to derive them (process diversity and process compliance), which indicates a good level of discriminant validity for them.

As our model specification contained three linear equations (equations 1-3) with potential contemporaneous cross-equation error correlations, we use seemingly unrelated regression (SUR) to estimate the model (Zellner 1962). We tested for endogeneity in the system of equations using Durbin-Wu-Hausman test (Davidson and MacKinnon 1993, pp.237-242).

\(^{10}\) We thank the review team for suggesting the different ways of deriving a process diversity variable to be used in the regression models.
Table 3. Summary Statistics (N=410)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td>0.16</td>
<td>0.26</td>
<td>0.01</td>
<td>3.37</td>
</tr>
<tr>
<td>Quality</td>
<td>0.02</td>
<td>0.03</td>
<td>0.71*10^-3</td>
<td>0.28</td>
</tr>
<tr>
<td>Process diversity†</td>
<td>0.28</td>
<td>0.26</td>
<td>0.00</td>
<td>0.91</td>
</tr>
<tr>
<td>Process compliance</td>
<td>0.16</td>
<td>0.11</td>
<td>0.01</td>
<td>0.42</td>
</tr>
<tr>
<td>Fit between process diversity and process compliance (deviation score)</td>
<td>-1.12</td>
<td>0.87</td>
<td>-3.12</td>
<td>-0.01</td>
</tr>
<tr>
<td>Fit between process diversity and process compliance (residual score)</td>
<td>0.03</td>
<td>1.17</td>
<td>-2.15</td>
<td>2.53</td>
</tr>
<tr>
<td>Agile KPAs score</td>
<td>27.14</td>
<td>24.67</td>
<td>2.27</td>
<td>41.54</td>
</tr>
<tr>
<td>Requirements volatility</td>
<td>15.58</td>
<td>22.27</td>
<td>0.00</td>
<td>275.05</td>
</tr>
<tr>
<td>Design and technology novelty</td>
<td>35.00</td>
<td>48.00</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Customer involvement</td>
<td>16.75</td>
<td>11.31</td>
<td>0.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Team size</td>
<td>10.86</td>
<td>10.48</td>
<td>2.00</td>
<td>116.00</td>
</tr>
<tr>
<td>Team experience</td>
<td>3.81</td>
<td>2.26</td>
<td>0.19</td>
<td>21.86</td>
</tr>
<tr>
<td>Code size</td>
<td>1467.34</td>
<td>2645.18</td>
<td>25.94</td>
<td>32767.00</td>
</tr>
<tr>
<td>Project manager certification</td>
<td>0.45</td>
<td>0.50</td>
<td>0.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Note: † Disparity process diversity score used in regression models; results are similar for the use of separation and variety process diversity scores.

Table 4. Correlations (N=410)

<table>
<thead>
<tr>
<th>Variable</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quality</td>
<td>0.18</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process diversity†</td>
<td>-0.16</td>
<td>-0.13</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process compliance</td>
<td>-0.11</td>
<td>-0.10</td>
<td>-0.33</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit (deviation)</td>
<td>0.15</td>
<td>0.08</td>
<td>0.32</td>
<td>0.38</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit (residual)</td>
<td>0.21</td>
<td>0.10</td>
<td>0.34</td>
<td>0.05</td>
<td>0.83</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Agile KPA score</td>
<td>-0.25</td>
<td>-0.16</td>
<td>0.53</td>
<td>-0.18</td>
<td>0.31</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements volatility</td>
<td>-0.04</td>
<td>0.04</td>
<td>0.24</td>
<td>-0.11</td>
<td>0.27</td>
<td>0.22</td>
<td>0.37</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design and technology novelty</td>
<td>0.10</td>
<td>0.13</td>
<td>0.54</td>
<td>-0.27</td>
<td>0.54</td>
<td>0.51</td>
<td>0.25</td>
<td>0.04</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer involvement</td>
<td>0.01</td>
<td>-0.10</td>
<td>0.55</td>
<td>-0.12</td>
<td>0.41</td>
<td>0.49</td>
<td>0.52</td>
<td>0.40</td>
<td>0.02</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team size</td>
<td>-0.06</td>
<td>-0.31</td>
<td>-0.11</td>
<td>-0.07</td>
<td>-0.01</td>
<td>-0.15</td>
<td>-0.23</td>
<td>0.01</td>
<td>0.18</td>
<td>-0.18</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team experience</td>
<td>-0.02</td>
<td>0.04</td>
<td>0.01</td>
<td>0.03</td>
<td>0.06</td>
<td>0.01</td>
<td>0.07</td>
<td>0.12</td>
<td>0.19</td>
<td>0.17</td>
<td>0.03</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code size</td>
<td>0.47</td>
<td>-0.17</td>
<td>0.15</td>
<td>-0.10</td>
<td>-0.07</td>
<td>-0.02</td>
<td>0.21</td>
<td>0.01</td>
<td>0.17</td>
<td>-0.04</td>
<td>0.57</td>
<td>0.01</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>Project manager certification</td>
<td>0.03</td>
<td>0.05</td>
<td>0.21</td>
<td>-0.12</td>
<td>0.21</td>
<td>0.18</td>
<td>0.19</td>
<td>0.07</td>
<td>0.23</td>
<td>0.04</td>
<td>0.04</td>
<td>0.27</td>
<td>0.03</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Note: All pairwise correlations >0.1 or < -0.1 are significant at p<0.05; disparity process diversity score used in regression models; results are similar for the use of separation and variety process diversity scores.

In all cases the Durbin-Wu-Hausman test indicated that the null hypotheses (variables are exogenous) could not be rejected, which indicates that the SUR regression results are
consistent\textsuperscript{11}. As an additional robustness check, we also performed a two-stage least squares instrumental variable regression to test the relationship between software process diversity, productivity, and quality\textsuperscript{12}. The two-stage least squares regressions yielded results consistent with SUR, lending support for the use of the SUR results for hypotheses testing.

Results pertaining to the antecedents of process diversity and utilized to verify hypotheses 1-4 are presented in Table 5. In Table 6, we present results pertaining to the two project performance variables, productivity and quality, estimated using the two different ways of measuring fit between process diversity and process compliance. We see that all the models are statistically significant at \(<1\%\) level and the adjusted R-squared values are indicative of good explanatory power. Post-regression diagnostics to detect outliers using Cook’s distance metric and leverage plots did not reveal any problems. The highest variance inflation factor among all the models was 2.6 and the highest condition index was 16.23, which confirm that multicollinearity issues are not of concern (Belsley \textit{et al.} 2004, p. 105).

<table>
<thead>
<tr>
<th>Independent Variables</th>
<th>Effect on Process Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process compliance</td>
<td>1 -0.087\textsuperscript{**} (0.012)</td>
</tr>
<tr>
<td>Requirements volatility</td>
<td>2 0.129\textsuperscript{***} (&lt;0.001)</td>
</tr>
<tr>
<td>Design and technology novelty</td>
<td>3 0.309\textsuperscript{***} (&lt;0.001)</td>
</tr>
<tr>
<td>Customer involvement</td>
<td>4 0.056\textsuperscript{***} (&lt;0.001)</td>
</tr>
<tr>
<td>Team size</td>
<td>5 -0.024\textsuperscript{**} (&lt;0.001)</td>
</tr>
<tr>
<td>Team experience</td>
<td>6 -0.146\textsuperscript{***} (&lt;0.001)</td>
</tr>
<tr>
<td>Code size</td>
<td>7 0.087\textsuperscript{***} (0.009)</td>
</tr>
<tr>
<td>Project manager certification</td>
<td>8 0.293\textsuperscript{***} (&lt;0.001)</td>
</tr>
<tr>
<td>Intercept</td>
<td>9 0.352 (0.210)</td>
</tr>
<tr>
<td>Sample size</td>
<td>410</td>
</tr>
<tr>
<td>Adjusted R-Squared</td>
<td>0.713</td>
</tr>
<tr>
<td>Chi-Squared</td>
<td>1112.20\textsuperscript{***} (&lt;0.001)</td>
</tr>
</tbody>
</table>

\textit{Note: Two-tailed p-values in parenthesis; \textsuperscript{**} significant at \(<1\%; \textsuperscript{***} \text{significant at \(<5\%; \textsuperscript{***} \text{significant at \(<10\%; \textsuperscript{† Disparity process diversity score used in regression model; results do not vary across the use of disparity, separation, and variety process diversity scores as well as their linear combinations.}

\textsuperscript{11} Durbin-Wu-Hausman test statistics were as following for the productivity and quality equation respectively: F(1, 399)=0.63, p>F =0.427; F(1, 399)=1.02; p>F=0.312.

\textsuperscript{12} We utilized instrumental variables used in prior studies, namely, defect density of in-process defects and the ratio of upfront project investments for uniquely identifying the two-stage least square regression models (Krishnan \textit{et al.} 2000; Harter \textit{et al.} 2000).
Referring to Table 5, we see that requirements volatility, design novelty, and customer involvement all had a positive and statistically significant effect on process diversity. Confirming hypothesis 1, the results indicate that a percentage increase in requirements volatility leads to about 13 percentage points increase in process diversity ($\beta=0.129; \ p<0.001$). Validating hypothesis 2, we see that a unit increase in design and technology novelty score increase process diversity by about 31 percentage points ($\beta=0.309; \ p<0.001$). As predicted by hypothesis 3, a percentage increase in customer involvement in the project increased process diversity in the
project by about 6 percentage points (β=0.056; p<0.001). Hypothesis 4 posited that an increase in process compliance efforts in a project would be negatively associated with process diversity. Our results support this prediction and show that a unit increase in process compliance efforts decreases process diversity within a project by about 9% percentage points (β=-0.087; p<0.05). Results for the other control variables show that larger teams and more experienced teams have lower levels of process diversity. In contrast, projects with a bigger code base and projects led by managers who were professionally certified in any of the five normative process frameworks adopted by the research site were associated with increased levels of software process diversity.

Recall that hypotheses H5a and H5b predicted that a better fit between process diversity and process compliance efforts yield higher productivity and quality respectively. To verify this prediction, we refer to the results pertaining to columns 1-4 in Table 6. Coefficients presented in columns 1 and 2 of Table 6 pertain to the regression specifications using the fit variable derived as a deviation score. Columns 3 and 4 present the regression coefficients when an alternate residual score measure is used to derive the fit variable. Overall, the regression results are consistent and do not vary irrespective of the way the fit between process diversity and process compliance is measured. The results confirm our hypotheses that a better fit between process diversity and process compliance efforts is associated with higher levels of project performance. We see that a unit increase in the fit between process diversity and process compliance improves productivity by 1.15 units (α=1.152; p<0.001; Table 6, column 1) and increases software quality by 0.31 units (γ=0.312; p<0.001; Table 6, column 2). The individual, direct effects of both process diversity and process compliance on productivity and quality are not statistically significant. An important insight from this empirical result is that the fit between process diversity and process compliance, and not those mechanisms acting independently, has an impact
on project performance. Therefore, in order to benefit from process diversity, concomitant investments in organizational process compliance activities are necessary.

Results for other control variables show that requirements volatility has a significant negative effect on productivity, but does not significantly affect quality. Based on this result we infer that project teams in our dataset encountered productivity losses for maintaining a higher level software quality when they faced client-driven requirements changes. We also find that bigger software code base and larger team size have a negative effect on project performance. Finally, we find that a higher degree of customer involvement is negatively associated with project performance. This implies that project teams we observed worked better when presented with well-specified and codified customer contracts that facilitated minimal intervention from customers. Although a higher level of customer involvement in a project is commonly expected to be positively associated with project success, prior research findings have shed light on some of the mixed effects of a high degree of customer involvement, such as the possibility of a conflict-ridden, lengthy, and less effective development process (Heinbokel et al. 1996; Subramanyam et al. 2010). It is possible that the presence of process diversity in a high process maturity environment exacerbates such negative effects of a higher degree of client involvement. Also, clients could have increased their involvement if they perceived their projects as not performing well. Beyond these speculations, we do not have a sufficient basis to tease out the causal direction of the effects of customer involvement. There is a need for further investigation of the effects of customer involvement on project performance in the presence of process diversity in high process maturity environments.

**Robustness Test for Group Size Effects**

Diversity variables operationalized to measure group-level separation, variety, and
disparity scores have been reported to suffer from biases when there is a large variation of group sizes in the sample (Biemann and Kearney 2010). Recall that we treat a KPA as the fundamental unit of a software process, and group size is determined by the membership of KPAs to process templates. To rule out group size-induced biases in our results, we specifically accounted for group size (number of KPAs) while deriving the variance and standard deviation scores as recommended in the literature (cf. Biemann and Kearney 2010). Regression estimates using the original and the bias-corrected process diversity scores were similar, indicating that group size-induced biases are not of a concern in this study.  

**DISCUSSION**

*Implications for Research*

This study takes an important step forward in rigorously conceptualizing, measuring, and analyzing software process diversity. We have laid the groundwork for theorizing software process diversity as an outcome of choices made by a project team in response to specific contingencies faced during the lifecycle of the project. By drawing on prior work on organizational and demographic diversity (cf. Harrison and Klein 2007), we conceptualized how software process diversity can be understood not only in terms of the traditional separation of plan versus agile dichotomies (i.e., the separation dimension), but also along the variety and disparity dimensions. Even when software projects appear homogenous at a surface level classification of plan versus agile, a consideration of the additional variety and disparity dimensions might suggest that there is greater heterogeneity, such as when project teams adopt and fuse different frameworks from within the agile or within the plan-based family of process frameworks. We believe that the study has laid a good foundation to “move beyond the entrenched disagreements about planning versus agility” (Austin and Devin 2009), and has

---

13 We thank the Senior Editor and Associate Editor for recommending this robustness check.
created a theoretically well-informed and empirically verifiable framework for studying software process diversity. We have established a rigorous case for usefully combining the disparate control and flexibility-focused process frameworks that could facilitate a new generation of software process innovations.

Our finding regarding the importance of a good fit between process diversity and process compliance has important implications. It suggests that as process diversity proliferates within a project, adequate and appropriate process compliance mechanisms are needed to mitigate hazards due to poor or ill-conceived process variants. This result calls for expanding the current theoretical conceptualizations of software project risk management to accommodate process variations within a project as a potential and distinct source of risk that spans both the social and technical subsystems of a project (Wallace et al. 2004; Iversen et al. 2004).

The need to achieve a good fit between software process diversity and process compliance has at least two additional implications for software project management. First, it is important to understand how adequate autonomy and appropriate coordination mechanisms could be instituted between project-level personnel and centrally organized (at firm-level) compliance personnel without exacerbating the tradeoffs between efficiency and flexibility (Ramasubbu 2014; Subramanyam et al. 2012). Second, prior conceptualizations of project control (e.g., Kirsch et al. 2002; Nidumolu and Subramani 2003) need to be complemented with new mechanisms that explicitly take into consideration the effects of process diversity. We only studied the overall effort expended on process compliance, leaving open the need for further investigations of how specific portfolios of controls could be designed and implemented to regulate the fit between process diversity and process compliance.

**Implications for Practice**

An important implication for practice that stems from this research is the need to
judiciously manage the alignment between process diversity and process compliance in software projects. Based on our discussions with the practitioners at our research site, we provide a few implementable guidelines for software development organizations. First, development of meta-routines or problem-solving procedures that are independent of specific normative process frameworks would provide firms with an organization-wide platform for improving the efficiency of compliance mechanisms without compromising overall flexibility (Adler et al. 1999). Such meta-routines need to be enabled through appropriate infrastructural and organizational support mechanisms such as automation of process template audits and adequate “separation of concerns” between organizational-level SEPG personnel and project-level development personnel (Ramasubbu 2014).

Second, we noticed that teams used scope and temporal partitioning for creating effective boundaries between the applications of different varieties of processes in their projects. The possibility of spatial partitioning, that is, the use of separate specialized units that could independently handle the distinct aspects of plan-based and agile processes were also raised during our discussions with project managers. The use of such scope, temporal, and spatial partitioning of tasks within a unit has been known to aid organizational ambidexterity (Puranam et al. 2006). In such scenarios, partitioning and switching events of a project need to be used to trigger the necessary process compliance investments. For example, whenever a software team performs process partitioning or switching, an automatic audit by SEPG could be triggered, thereby facilitating a systematic way to match the levels of process diversity and process compliance in a software project. Furthermore, process partitioning and switching milestones could be used for enrichment and learning activities such as peer-reviews, root-cause analyses, and formal training that improve software project performance (Ramasubbu et al. 2008). Thus,
we recommend SEPG managers to develop infrastructural capabilities that help implement mechanisms to track and trace project-level process partitioning and switching events.

Finally, since our results indicated that customer involvement plays an important role in the way project teams initiate process variations, we recommend SEPG personnel to use cues from customer communication at the organizational-level (e.g., customer satisfaction surveys) to triangulate and verify the actual need to approve or block process variations. Such triangulation of customer feedback from multiple communication channels (organizational and project levels) could help process compliance personnel to identify and rectify the existence of contradictory customer policies at different levels, and, thereby, avoid unnecessary or risky process variations.

**Limitations and Further Research**

There are some limitations of this study that future research could address. First, we could not fully distinguish the unique effects of separation, variety, and disparity dimensions of software process diversity in this research. The three diversity scores were highly correlated in our dataset, and we suspect that idiosyncratic characteristics of the development environment at our research site could be a reason for this. Future research could shed light on the factors that lead to distinct effects of the separation, variety, and disparity process diversity dimensions through data gathered from a broader set of firms. Second, since we observed only custom (bespoke) software development projects, we should be cautious in generalizing our results across all types of software development projects (maintenance, reengineering, product development, etc.). The conceptual model and empirical analysis utilized in this study can be replicated in other project settings and future research could embark on the necessary comparative analyses. Third, we did not observe the long-term impacts of process diversity, and only studied its impact on immediate project performance outcomes. Further research is needed to ascertain the long-term impacts of software process diversity on learning curves and capability.
development of project teams. Fourth, we studied only software projects that used standard process components drawn from well-established normative process frameworks. Not all software teams utilize standard process frameworks, and we need to be cautious in extrapolating the results reported in the paper to teams that use a variety of home-grown processes. Finally, the distinct coexistence of several process designs in a software production ecosystem for longer periods of time, and the way those diverse process designs coevolve and adapt to each other warrants further examination. We believe that these are fruitful lines of enquiry for future research on software process diversity.

REFERENCES


