Factors of Software Process Improvement Success in Small and Large Organizations: An Empirical Study in the Scandinavian Context

Tore Dybå
SINTEF Telecom and Informatics
S.P. Andersens vei 15, NO-7465 Trondheim, Norway
Phone: +47 73 59 29 47, Fax: +47 73 59 29 77
tore.dyba@sintef.no

ABSTRACT
Existing software engineering and organization development literature acknowledges that there are fundamental operational differences between small and large organizations. Despite this recognition, there has been no attempt to verify whether small and large software organizations implement software process improvement (SPI) programs differently in order to advance their businesses.

This study examines whether an organization’s size affects its SPI implementation strategy and the degree of SPI success. Based on an extensive literature review of critical factors of quality management, organizational learning, and SPI, a survey questionnaire was developed and data on the implementation of six organizational factors and the resulting organizational performance was collected through a mail survey of 120 software organizations. The findings show that small organizations reported that they implement SPI elements as effectively as large organizations, and in turn, achieve high organizational performance. The main lesson to be learned from this study is that small software organizations should capitalize on their relative strengths in employee participation and exploration of new knowledge.

Categories and Subject Descriptors: [D2.8 Software Engineering]: Metrics–performance measures; process metrics

General Terms: Management, measurement

Keywords: Software process improvement, learning software organization, critical success factors, survey research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1 ESEC/FSE’03, September 1–9, 2003, Helsinki, Finland
COPYRIGHT 2003 ACM 1-58113-743-5/03/0009...$5.00.

1. INTRODUCTION
Knowledge is the most decisive factor in software development. Advancement strategies, thus, depend on learning – the creation of new knowledge. Such knowledge creation is not simply a matter of compiling facts; it is rather a dynamic human process that is related to the practice of individuals and groups.

In effect, project teams can be considered as the fundamental learning unit in modern software organizations. Teams often develop a common set of routines, priorities, and approaches through dialogue and discussion that ultimately produce important new insights. However, the software team is seldom concerned with how the new insights might be transferred to people outside the team – to the rest of the organization.

Software processes play an important role here, in coordinating different teams in large organizations so that their practices don’t grow out of touch with one another. Ideally, these processes should combine the need for rigor and discipline with the need for flexibility and creativity, but that balance is hard to achieve [14]. Formal processes emphasize the explicit command-and-control side of the organization, while informal team practices emphasize the mutual adjustment and explorations needed to get work done. Practice without process tends to become unmanageable; process without practice results in the loss of creativity needed for sustained innovation [9]. Accordingly, there is an inherent dilemma of balancing the tension between formal process and the informal, social practice for anyone, whether large or small, engaged in SPI and in developing SPI methods.

Based on these considerations, existing software engineering and organization development literature acknowledges that there are fundamental operational differences between small and large organizations (see e.g. the Sept./Oct. 2000 issue of IEEE Software). Small organizations...
seem more concerned about practice, while large organizations seem more concerned about formal process. Despite this recognition, there has been no systematic attempt to verify whether small and large software organizations implement software process improvement (SPI) programs differently in order to advance their businesses.

There is, therefore, a need to determine what differences, if any, exist between small and large software organizations in SPI implementation strategy and the degree of success. Thus, the major objectives of this study are to investigate whether:

1. Large software organizations use the same SPI strategies as small software organizations;
2. Large software organizations yield higher performance than small software organizations;
3. Successful organizations (both small and large) implement SPI strategies more effectively than the less successful organizations; and
4. Large successful software organizations use different SPI strategies than small successful software organizations.

This study is based on an extensive literature review within the fields of organization development and change, organizational learning, quality management, and SPI combined with a multiple case study of four software organizations and a survey of SPI experts from both industry and academia [12]. From these investigations, a survey questionnaire was developed and data on the implementation of six organizational factors and the resulting organizational performance was collected. Thus, these literature-based and empirically tested critical factors of SPI formed the basis for the analysis reported in this article.

2. RESEARCH METHOD
2.1 Population and sample
Software intensive organizations were considered as the target population for this study. This population includes companies of different sizes, developing either software or combined software and hardware products for a wide variety of markets.

Software managers and quality managers in the Norwegian IT industry with corporate membership either in the Association of the Norwegian Software Industry (PROFF) or the Norwegian IT Technology Forum (ITUF) were chosen as the sampling frame for the survey. Both organizations are active in the area of software development and taken together, they were considered to be representative for software development within the Norwegian IT industry. Since the unit of analysis in this study was the software organization, the managers were asked to answer on behalf of their respective organizations.

Sample size might have a substantial impact on all results in quantitative research. The effects of sample size are seen most directly in the statistical power of the significance testing and the generalizability of the results. Sample size is also an important consideration in the discussion of the internal consistency of items used in measurement scales. Given these considerations, the target sample size for ensuring adequate statistical power, generalizability, and item analysis in this study was a minimum of 100 respondents.

A random sample of 154 software and quality managers from the membership lists of the two associations were contacted by telephone to request participation in the study prior to mailing the questionnaires. All managers agreed to participate in the study. We provided the respondents with self-addressed, stamped return envelopes. Also, by keeping the questionnaire as short as possible (a pilot study of the survey questionnaire showed that respondents needed about 10 minutes to complete it), we combined several well-proven techniques for improving the response rate of mailed questionnaires.

A total of 120 software and quality managers representing whole organizations or independent business units within 55 software companies completed and returned the questionnaire. This is within the limits discussed above for both adequate statistical power and generalizability of the results in this study. Furthermore, this represents an effective response rate of 77.9 percent, which is well within the norm of 60 +/- 20 percent for representatives of organizations and mid-level managers suggested for academic studies by Baruch [5], and considerably higher than prior mail surveys conducted in the Norwegian software and IT industry. Given the high response rate in this study, no further analysis was done on the differences between respondents and non-respondents.

The respondent’s companies represented a wide variety of industry sectors. To a large extent, the organizations developed software for external customers (approximately 96% of the sample). Furthermore, the sample showed a mix of both small and large organizations, with approximately one third (37.5%) of the organizations having 30 or less developers and approximately one third (36.7%) having 200 or more. A large majority of the respondent’s organizations had a quality system in use (71.7%). The average length of the respondents’ job tenure at their current organization was 8.4 years, while professional tenure (years in software development) was 11.4 years. Two thirds of the respondents (65.9%) held a master’s or doctoral degree [12].
2.2 Variables and measures
The combined result of the literature review, the multiple case studies and the expert review was the identification and operationalization of six independent, one dependent, and two moderating variables.

Independent variables. The following six independent variables were identified and operationalized:

- Business orientation: the extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies.
- Involved leadership: the extent to which leaders at all levels in the organization are genuinely committed to and actively participate in SPI.
- Employee participation: the extent to which employees use their knowledge and experience to decide, act, and take responsibility for SPI.
- Concern for measurement: the extent to which the software organization collects and utilizes quality data to guide and assess the effects of SPI activities.
- Exploitation: the extent to which the software organization is engaged in the exploitation of existing knowledge.
- Exploration: the extent to which the software organization is engaged in the exploration of new knowledge.

To measure the extent to which each of these six independent variables were practiced, we used multi-item, five-point, bipolar Likert scales that ranged from “strongly disagree” (1) to “strongly agree” (5) for all indicators. The item ratings were summarized to form a summated rating scale for each independent variable.

Dependent variable. SPI success was operationalized and measured based on two multi-item measures. Each manager was asked to rate, on 5-point bipolar Likert scales, (1) the level of perceived SPI success and (2) the performance of their organization for the past three years with respect to cost reduction, cycle time reduction, and customer satisfaction. Two items were used to measure the level of perceived SPI success, while three items were used to measure organizational performance. The ratings for the two performance dimensions were averaged to form a single measure of overall SPI success.

Moderating variables. Two moderating variables – environmental conditions and organizational size – were operationalized and included in the study. Environmental conditions were measured using two semantic differential items, which were rated on a seven-point, bipolar, adjectival, graphic scale. The two items were stable vs. unstable environment and predictable vs. unpredictable environment. Organizational size was defined as the number of software developers in the organization.

Appendix A shows the summated rating scale items for the independent and dependent variables along with the corresponding reliability measures (Cronbach’s α) [10].

2.3 Data collection procedure
The questionnaire consisted of two parts: the first part asked for manager’s ratings of their software department with respect to factors of SPI success and organizational performance; the second part asked for general background information and an assessment of the environment.

In the first part, 36 separate items were used to measure the six factors of SPI success, while five items was used to measure the performance construct. Thus, each manager generated seven scores; one for each of the critical factors, and one for the success measure. All scales in this research had 5 to 8 items.

In the second part of the questionnaire, each manager assessed their organization’s environment using two semantic differential items, in addition to provide general background and demographic information.

2.4 Data analysis techniques
The main concern for the analysis was to differentiate the successful from the less successful software organizations and to explore the relationships between small and large organizations. Thus, we defined contrasted criterion groups based on the upper and lower third of the organizational size and performance distributions (see Figure 1). This is similar to the approach used by Hoch et al. [16] to learn which factors make the difference between success and failure in the software industry at large. The advantage of this approach, is a much sharper differentiation between the contrasted criterion groups that otherwise would be made more diffused by borderline cases. The major disadvantage, however, is the exclusion of one third of the data points, and thus lower statistical power.

We used parametric statistics such as the Pearson product-moment correlation coefficient, the t-test, the F statistic, and multiple regression analysis to analyze the data in this study. An important prerequisite, however, is that the assumptions of these analyses be met.

The results of testing these assumptions showed that all variables were within the acceptable range for the normal distribution assumption, and that the assumptions of homoscedasticity, linearity, and independence of the error terms were supported. Furthermore, no influential observations were identified, and the investigations of collinearity and multicollinearity indicated no problems. In other words, there were no extreme violations to the basic assumptions underlying the chosen data analysis techniques that could justify the use of less powerful non-parametric statistics.
3. RESULTS

To examine whether a software organization’s size affects its critical success factors and the resulting performance, we compared small (≤ 30 developers, N = 45) and large (≥ 200 developers, N = 44) organizations. The effects of organizational size were examined by testing the differences of the means between the small and large organizations using two-tailed t-tests. Results of these tests are summarized in Table 1. For each construct the table provides the mean score, standard deviation, and t-value.

Five of the six independent variables showed a statistically significant difference between large and small software organizations. Large organizations reported a significantly greater extent of business orientation (t = 2.71, p < 0.01), concern for measurement (t = 2.61, p < 0.05), and involved leadership (t = 1.79, p < 0.1) than small organizations. However, small organizations reported a highly significantly greater extent of exploration of new knowledge (t = -3.00, p < 0.005) and a significantly greater extent of employee participation (t = -1.75, p < 0.1) than large organizations. The last independent variable, the extent of exploitation of existing knowledge, were similar in large and small organizations. Furthermore, with respect to organizational performance, large software organizations reported a significantly higher overall SPI success (t = 2.18, p < 0.05) than small organizations.

To further examine the effects of organizational size, we compared the responses from large successful software organizations with the responses from large less successful organizations. These results are tabulated in Table 2. The comparison revealed that with the exception of exploration of new knowledge, large successful organizations reported executing all of the activities defined by the independent variables to a significantly greater extent than large less successful organizations.

Similarly, we made a comparison between small successful software organizations and small less successful organizations. As shown in Table 3, small successful organizations reported executing all of the activities defined by the independent variables to a highly significantly (p < 0.0005) greater extent than small less successful organizations. Furthermore, comparing these results with the results of the large organizations, we observe that the differences of the means between the large organizations are smaller than the corresponding differences of the means between the small organizations.

Finally, we compared small and large software organizations that were successful. Table 4 shows the differences that appear to exist between small and large organizations having SPI success. As can be seen, both small and large successful organizations reported the same level of success. Statistically significant differences were observed for two of the six independent variables: Small successful organizations reported higher levels of employee participation (t = -2.44, p < 0.05) and exploration of new knowledge (t = -3.50, p < 0.001) than the larger organizations.
Table 1. Large versus small software organizations.

<table>
<thead>
<tr>
<th></th>
<th>Large (n = 44)</th>
<th>Small (n = 45)</th>
<th>(t)-value(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPI success factors:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Business Orientation</td>
<td>3.48 .53</td>
<td>3.10 .77</td>
<td>2.71**</td>
</tr>
<tr>
<td>2. Involved leadership</td>
<td>3.68 .68</td>
<td>3.40 .75</td>
<td>1.79(^a)</td>
</tr>
<tr>
<td>3. Employee participation</td>
<td>3.41 .50</td>
<td>3.60 .49</td>
<td>-1.75(^a)</td>
</tr>
<tr>
<td>4. Measurement</td>
<td>3.42 .53</td>
<td>3.10 .61</td>
<td>2.61(^#)</td>
</tr>
<tr>
<td>5. Exploitation</td>
<td>3.39 .55</td>
<td>3.30 .63</td>
<td>.67</td>
</tr>
<tr>
<td><strong>Performance:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall SPI success</td>
<td>3.63 .52</td>
<td>3.39 .51</td>
<td>2.18(^*)</td>
</tr>
</tbody>
</table>

\(^{\#} p < 0.06\) \(^{\pi} p < 0.01\) \(^{\ast} p < 0.05\) \(^{\ast \ast} p < 0.01\) \(^{\ast \ast \ast} p < 0.005\)

1. All \(t\)-tests are two-tailed.

Table 2. Large successful versus large less successful software organizations.

<table>
<thead>
<tr>
<th></th>
<th>Large successful ((n = 20))</th>
<th>Large less successful ((n = 11))</th>
<th>(t)-value(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPI success factors:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Business Orientation</td>
<td>3.80 .46</td>
<td>3.16 .45</td>
<td>3.71**</td>
</tr>
<tr>
<td>2. Involved leadership</td>
<td>3.97 .62</td>
<td>3.49 .71</td>
<td>1.95(^g)</td>
</tr>
<tr>
<td>3. Employee participation</td>
<td>3.69 .49</td>
<td>3.12 .49</td>
<td>3.11*</td>
</tr>
<tr>
<td>4. Measurement</td>
<td>3.72 .49</td>
<td>3.23 .41</td>
<td>2.82(^a)</td>
</tr>
<tr>
<td>5. Exploitation</td>
<td>3.71 .42</td>
<td>2.93 .42</td>
<td>4.97***</td>
</tr>
<tr>
<td>6. Exploration</td>
<td>3.39 .50</td>
<td>3.12 .52</td>
<td>1.44</td>
</tr>
</tbody>
</table>

\(^g p = 0.06\) \(^\pi p < 0.01\) \(^\ast p < 0.005\) \(^\ast \ast p < 0.001\) \(^\ast \ast \ast p < 0.0005\)

1. All \(t\)-tests are two-tailed.

Table 3. Small successful versus small less successful software organizations.

<table>
<thead>
<tr>
<th></th>
<th>Small successful ((n = 12))</th>
<th>Small less successful ((n = 17))</th>
<th>(t)-value(^1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPI success factors:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Business Orientation</td>
<td>3.85 .47</td>
<td>2.48 .60</td>
<td>6.63***</td>
</tr>
<tr>
<td>2. Involved leadership</td>
<td>4.10 .34</td>
<td>2.88 .58</td>
<td>6.49***</td>
</tr>
<tr>
<td>3. Employee participation</td>
<td>4.08 .37</td>
<td>3.20 .37</td>
<td>6.36***</td>
</tr>
<tr>
<td>4. Measurement</td>
<td>3.65 .60</td>
<td>2.84 .50</td>
<td>3.97***</td>
</tr>
<tr>
<td>5. Exploitation</td>
<td>3.87 .45</td>
<td>2.87 .52</td>
<td>5.38***</td>
</tr>
<tr>
<td>6. Exploration</td>
<td>4.01 .46</td>
<td>3.29 .46</td>
<td>4.12***</td>
</tr>
</tbody>
</table>

\(^{***} p < 0.0005\)

1. All \(t\)-tests are two-tailed.
firms, small firms relied on an implementation strategy that implemented TQM elements at least as effectively as large firms. Golhar [2] found that, while small successful firms compared large and small firms. For example, research on organization size [15], [21] and on quality management literature that ideal quality management should not be affected by contextual variables (e.g. [17]), this findings suggests that it does, indeed, when it comes to implementation issues. This is consistent with recent studies that demonstrated that small organizations can achieve comparable levels of overall SPI success with large organizations regardless of the environment. Moreover, they found that the SMEs approach to BPR was less rigorous, more resembling a socio-technical approach.

Our findings regarding the differences between large and small organizations are also consistent with the views expressed by Mohamed Fayad’s, Mauri Laitinen’s, and Robert Ward’s “Thinking Objectively” column in Communications of the ACM (1997-2001) regarding software engineering and SPI in small organizations. Similar views were also presented in the special section of the September/October 2000 issue of IEEE Software on software engineering in the small [19].

Thus, contrary to the claims put forward in the quality management literature that ideal quality management should not be affected by contextual variables (e.g. [17]), these findings suggest that it does, indeed, when it comes to implementation issues. This is consistent with recent research on organization size [15], [21] and on quality management comparing large and small firms. For example, in a survey of 488 U.S. and Canadian firms, Ahire and Gohar [2] found that, while small successful firms implemented TQM elements at least as effectively as large firms, small firms relied on an implementation strategy that involved significantly higher levels of employee involvement and flexibility than the larger firms’ strategy.

Also, in a study of Business Process Reengineering (BPR) in 134 Canadian enterprises, Raymond et al. [24] found that the small and medium-sized enterprises (SMEs) perceived an overall level of advantages equal to that of the large enterprises. Moreover, they found that the SMEs approach to BPR was less rigorous, more resembling a socio-technical approach.

Our findings regarding the differences between large and small organizations are also consistent with the views expressed by Mohamed Fayad’s, Mauri Laitinen’s, and Robert Ward’s “Thinking Objectively” column in Communications of the ACM (1997-2001) regarding software engineering and SPI in small organizations. Similar views were also presented in the special section of the September/October 2000 issue of IEEE Software on software engineering in the small [19]. Also by Brodman and Johnson [8] and Batista and de Figueiredo [6] regarding the implementation of CMM in small organizations.

4. DISCUSSION
The findings from this study showed that there are fundamental differences between small and large software organizations. With respect to organizational performance in general, large software organizations reported a higher level of overall SPI success ($t = 2.18, p < 0.05$) than small organizations. This is consistent with e.g. Terziovski and Samson’s [26] finding that larger companies tend to gain greater benefits from Total Quality Management (TQM) than smaller firms. This is also as expected, given that an organization will grow larger with repeated success. However, when comparing small successful organizations with large successful organizations, there was no difference in the level of success.

What’s more important, however, is that large successful and small successful organizations differ fundamentally in their respective approach to SPI, specifically with respect to participation and the preferred mode of learning. As can be seen from Table 4, small successful organizations reported higher levels of employee participation ($t = -2.44, p < 0.05$) and exploration of new knowledge ($t = -3.50, p < 0.001$) than the larger organizations.

Thus, contrary to the claims put forward in the quality management literature that ideal quality management should not be affected by contextual variables (e.g. [17]), these findings suggest that it does, indeed, when it comes to implementation issues. This is consistent with recent research on organization size [15], [21] and on quality management comparing large and small firms. For example, in a survey of 488 U.S. and Canadian firms, Ahire and Gohar [2] found that, while small successful firms implemented TQM elements at least as effectively as large firms, small firms relied on an implementation strategy that involved significantly higher levels of employee involvement and flexibility than the larger firms’ strategy.

Also, in a study of Business Process Reengineering (BPR) in 134 Canadian enterprises, Raymond et al. [24] found that the small and medium-sized enterprises (SMEs) perceived an overall level of advantages equal to that of the large enterprises. Moreover, they found that the SMEs approach to BPR was less rigorous, more resembling a socio-technical approach.

Our findings regarding the differences between large and small organizations are also consistent with the views expressed by Mohamed Fayad’s, Mauri Laitinen’s, and Robert Ward’s “Thinking Objectively” column in Communications of the ACM (1997-2001) regarding software engineering and SPI in small organizations. Similar views were also presented in the special section of the September/October 2000 issue of IEEE Software on software engineering in the small [19]. Also by Brodman and Johnson [8] and Batista and de Figueiredo [6] regarding the implementation of CMM in small organizations.

Also, it is interesting to note that, as there was no difference in the level of exploitation between small and large organizations regardless of the environment, there was a marked difference in the level of exploration. The results showed that small software organizations engaged in significantly ($p < 0.001$) more exploration in turbulent environments than large software organization, see [13]. This suggests that the main difference between small and large software organizations is the ways in which they react to unstable and changing stimulus situations. In small organizations, the response to a changed situation was an increased level of exploration. This supports the assertion that small software organizations in turbulent environments
require learning strategies that are more closely aligned with explorative behavior, while at the same time promoting the exploitation of past experiences. As discussed in [13], this is at the heart of an SPI strategy based on improvisation.

However, while most software managers agreed that changes in their competitive environment are fast and increasingly unpredictable, managers of large software organizations still relied on learning from experience to prepare for the future rather than exploring new possibilities. They tended to generate the same response even when the stimuli had changed. They kept doing what they did well, rather than risk failure. This behavior is supported by the organizational literature, which suggests that large organizations are less likely to change in response to environmental changes than small organizations. There could be many reasons for this mode of learning. Tushman and Romanelli [27], for example, argued that increased size leads to increased complexity, increased convergence, and thus, increased inertia.

Likewise, Mintzberg [23] postulated that the larger an organization is, the more formalized is its behavior. So, while small organizations can remain organic, large organizations develop bureaucracies with job specialization and sharp divisions of labor, emphasizing stability, order, and control. As a consequence, they often have great difficulties in adapting to changing circumstances because they are designed to achieve predetermined goals – they are not designed for innovation.

From a learning perspective, however, inertia develops as a result of the organization’s performance history [20]. Large organizations tend to be successful since an organization will grow larger with repeated success. However, since success reduces the probability of change in a target-oriented organization [11], large software organizations will be less likely to change when the environment changes.

Another explanation, which is also a direct consequence of using “best practice” models, is the institutionalized routines. In stable situations, such routinization can become an effective way of developing software, but it can also drive out the exploration of new alternative routines. The deeper these routines are grounded in the organizational culture, the more difficult they are to change and more easily they turn into an obstacle to improvement.

A further explanation is that the “best practice” approach to software development considers failure as unacceptable. This is consistent with the goal of promoting stability and short-term performance, as is the case in exploitation. In this situation, success provides an excellent foundation for increased reliability. On the other hand, success tends to encourage the maintenance of the status quo by generating single-loop learning. Failure, however, can generate double-loop learning, in which people question the assumptions behind their failures as well as the failures themselves [3]. The absence of failure experiences can, therefore, result in decreased organizational competence when faced with changing and turbulent environments.

Therefore, it is important to note that successful SPI requires tolerance for failure, and that failure is an essential prerequisite both for learning and for challenging the status quo. Software businesses, whether small or large, must thus be able to turn unexpected problems and failure into learning opportunities [1].

5. CONCLUSIONS

Based on the results from this study, it is reasonable to conclude that there exist a few factors that have a significant and positive effect on software business performance in both large and small organizations. Even without formally launching an SPI program, it appears that a software organization can achieve high organizational performance by executing the six identified constructs to their fullest extent. This means that software organizations of any size can advance their business by practicing a critical set of SPI elements. In other words, the size of the organization does not limit its potential for SPI success.

In summary, our findings show that small organizations reported that they can and do implement SPI elements as effectively as large organizations, and in turn, achieve high organizational performance. This indicates that SPI can be used as a competitive advancement strategy for both small and large software organizations.

However, a general observation from this study has been that large successful software organizations emphasize exploitation of their “best practices” through formal procedures, process models, guidelines, rules, check-lists etc. in order to manage and improve their software processes. Small successful organizations, on the other hand, put more emphasis on exploring new possibilities, making the most of the diversity and creativity of the human resources involved in software projects.

The lesson to be learned from this is that to implement SPI at least as effectively as their large counterparts, small software organizations should capitalize on their relative strengths in employee participation and exploration of new knowledge. This implies that small software organizations require learning strategies that are closely aligned with explorative behavior, while at the same time promoting the exploitation of past experience. Thus, to be successful, our findings suggest that formal processes must be supplemented with informal, inter-personal coordination about practice. This is consistent with Kraut and Streeter’s [18] conclusions on the subject of coordination in software development and with Teasley et al.’s [25] study on the collocation of software teams. Process and practice, then, do not represent rival views of the organization. Rather,
they reflect the creative tension at the center of innovative organizations.” [9].

Furthermore, the inherent lack of conformance between formal process and informal practice is a necessary feature, so that the software organization may learn from its theories of action. Deeper, second-order or double-loop learning not only requires that the software organization takes actions to correct errors, but also that it critically reexamines the validity of its basic norms and assumptions and, as a result, is prepared to modify existing strategies.

Rather than detailed process descriptions, this suggests that software processes should be described according to the “rule of bifurcation” [4], supporting different degrees of formality. I.e., they could be stated in terms of two levels: a general description of the process, supplemented by specific examples from practice. Like the tale of the sitar player asking Buddha how best to tune his instrument, therefore, we find the famous answer: “not too tight, and not too loose” as a good general rule for balancing formal process with informal practice.

5.1 Limitations

Although we have discussed several implications of our results, the research reported here is not without its limitations. First, the results have been discussed as though the facilitating factors caused SPI success. However, the investigation was cross-sectional and these assumptions of causality are not technically justified. It is possible, for example, that there are complex feedback mechanisms by which performance in one time period is affected by performance in previous periods [22]. Both success and lack of success might very well be self-reinforcing. On the other hand, there could also be negative feedback mechanisms by which SPI success or failure creates countervailing tendencies. It is possible therefore that performance below target levels increases organizational efforts and, thus, the likelihood of subsequent success. Accordingly, success can also be seen as a trigger for adjustments in the opposite direction through decreased innovation or increased slack. Therefore, many of the facilitating factors that seem likely to influence SPI success can themselves be influenced by prior success or lack of success. Finding the “true” causal structure of SPI success based on incomplete information generated by prior experience is, therefore, problematic.

Second, although generally accepted psychometric principles were used to develop the measurement instrument, the variables were still measured on the basis of subjective performance definitions. Performance measures such as the return on investment, net present value, and payback periods are often regarded as objective measures. However, attempts to provide objective definitions of such measures may be as open to criticism as subjective definitions [7]. This points to a general problem of defining and measuring success in studies of SPI. The question, therefore, is not whether such measures are subjective or not, but what purpose they serve.

Finally, a further complication is that the independent variables were assessed using retrospective recall. This involves a risk for the introduction of retrospective bias. It is possible, therefore, that performance information itself colors subjective memories and perceptions of possible causes of SPI success or failure. Software organizations are constantly worried about their performance, and “common wisdom” has many explanations for good and poor performance. As a result, retrospective reports of independent variables may be less influenced by memory than by a reconstruction that connects common wisdom with the awareness of performance results [22].

Despite these limitations, this study contributes to the growing literature on empirical software engineering research and provides empirical support for the importance of organizational issues and contextual variables in SPI.

5.2 Future work

A central problem of traditional software engineering is that it does not separate the tasks needed to coordinate large groups from those inherent for building software. Since the overwhelming majority of companies developing software are small, future studies should focus on the specific needs of small software organizations in more depth; for example, through longitudinal, multiple case studies.

Also, research should be related to the study of new and improved measures of SPI success, comparison of measurement instruments, and validation of SPI success measures. In addition, research is needed to investigate the importance of contextual variables to several types of SPI problems and to validate the approaches proposed for solving them. This way, we can tailor software engineering methods and improvement strategies to better help small, as well as large, organizations succeed with their unique business of software.

6. ACKNOWLEDGEMENTS

This work was supported in part by the Research Council of Norway under grant no. 118206/221 and by SINTEF Telecom and Informatics. The author would like to thank all respondents of the survey investigation for their willingness to participate in the inquiries.
7. REFERENCES
APPENDIX A: Measurement Instrument

A more detailed analysis and discussion of the validity and reliability of the operationalized measures for the variables in this study can be found in [12].

Business Orientation ($\alpha = 0.81$)
1. We have established unambiguous goals for the organization’s SPI activities.
2. There is a broad understanding of SPI goals and policy within our organization.
3. Our SPI activities are closely integrated with software development activities.
4. Our SPI goals are closely aligned with the organization’s business goals.
5. We have a fine balance between short-term and long-term SPI goals.

Leadership Involvement ($\alpha = 0.87$)
6. Management is actively supporting SPI activities.
7. Management accepts responsibility for SPI.
8. Management considers SPI as a way to increase competitive advantage.
9. Management is actively participating in SPI activities.
10. SPI issues are often discussed in top management meetings.

Employee Participation ($\alpha = 0.80$)
11. Software developers are involved to a great extent in decisions about the implementation of their own work.
12. Software developers are actively contributing with SPI proposals.
13. Software developers are actively involved in creating routines and procedures for software development.
14. We have an on-going dialogue and discussion about software development.
15. Software developers have responsibility related to the organization’s SPI activities.
16. Software developers are actively involved in setting goals for our SPI activities.
17. We have an on-going dialogue and discussion about SPI.

Concern for Measurement ($\alpha = 0.81$)
18. We consider it as important to measure organizational performance.
19. We regularly collect quality data (e.g. defects, timeliness) from our projects.
20. Information on quality data is readily available to software developers.
21. Information on quality data is readily available to management.
22. We use quality data as a basis for SPI.
23. Our software projects get regular feedback on their performance.

Exploitation of Existing Knowledge ($\alpha = 0.78$)
24. We exploit the existing organizational knowledge to the utmost extent.
25. We are systematically learning from the experience of prior projects.
26. Our routines for software development are based on experience from prior projects.
27. We collect and classify experience from prior projects.
28. We put great emphasis on internal transfer of positive and negative experience.

Exploration of New Knowledge ($\alpha = 0.85$)
29. We are very capable at managing uncertainty in the organization’s environment.
30. In our organization, we encourage innovation and creativity.
31. We often carry out trials with new software engineering methods and tools.
32. We often conduct experiments with new ways of working with software development.
33. We have the ability to question “established” truths.
34. We are very flexible in the way we carry out our work.
35. We do not specify work processes more than what is absolutely necessary.
36. We make the most of the diversity in the developer’s skills and interests to manage the variety and complexity of the organization’s environment.

Organizational Performance ($\alpha = 0.76$)
1. Our SPI work has substantially increased our software engineering competence.
2. Our SPI work has substantially improved our overall performance.
3. Over the past 3 years, we have greatly reduced the cost of software development.
4. Over the past 3 years, we have greatly reduced the cycle time of software development.
5. Over the past 3 years, we have greatly increased our customer’s satisfaction.