# SOFTWARE PROCESS IMPROVEMENT DEPLOYMENT: AN EMPIRICAL PERSPECTIVE

**JEAN-PIERRE KUILBOER**

UNIVERSITY OF MASSACHUSETTS

**NOUSHIN ASHRAFI**

UNIVERSITY OF MASSACHUSETTS

## ABSTRACT

Some experts in software development look upon Software Process Improvement (SPI) as the key to product quality at a reduced cost. Practitioners, on the other hand, have been slow in adopting SPI, and would argue that SPI methodologies are expensive, time consuming, and have little impact on software quality. The literature is full of anecdotal evidence supporting or opposing the benefits of SPI methodologies; yet, there is little empirical data to validate or contradict them. Whether it is the image created by complying with established guidelines or actual quality improvement that gains customer satisfaction and trust is to be determined. We conducted a survey in the New England region to get a perspective on these issues from practitioners in the field. Altogether, 67 responses were collected, which included developers who use some sort of SPI methodology, and those who do not. The purpose of this paper is to get an empirical perspective on the deployment of SPI techniques and their impact on product quality and productivity. Our survey was designed for software developers, and its questions may help us identify factors that influence the adoption of a software process improvement technique. According to the majority of our respondents, while using SPI does not necessarily lead to a quality software product at a reduced cost or delivery time, it creates a perception of quality leading to customer satisfaction.

## INTRODUCTION

The ultimate goal of software development is to improve customer satisfaction by delivering high-quality software in a timely manner. Achieving this goal, however, has proved to be a difficult and frustrating endeavor. Recent studies show that most software products are of poor quality, often late, and frequently exceed budget costs. In the last two decades, many tools and techniques for software improvement, such as CASE tools, 4GL, structured techniques, rapid prototyping, and software reengineering, have been introduced. Each has gone through its period of glory and been considered the answer to the software development crisis, but then has failed to deliver what was expected. One by one, each effort has lost momentum.

Over time, focus has shifted between improving the end product and process improvement. During the late 1970's and through the early 1980's, defining and improving software quality was the major challenge. Starting late in the 1980's and continuing through the 1990's, process improvement became the focus of attention, with the supporting argument that, "the quality of a software system is governed by the quality of the process used to develop it." Current efforts, however, are returning to an emphasis on quality in terms of the functional usefulness of the final product.

A look at the definition of the software process, Software Process Improvement (SPI), and some SPI initiatives that have been most commonly used, will help us obtain a greater understanding of the arguments for and against SPI methodologies.

The software development 'process' could be defined as all activities performed to develop or maintain a software product. Software Process Improvement is

about introducing changes to the software development process with the purpose of meeting a set of criteria, which promote process performance within quality, cost, and schedule targets. There are a number of process improvement initiatives, of which the most commonly used are the Software Engineering Institute's Capability Maturity Model (CMM), and the International Organization for Standardization's ISO-9000. These practices provide a set of guidelines for the planning, managing, tracking, and oversight of software development and maintenance.

**CMM and ISO Guidelines**

The concept of statistical repeatability, established by W.E. Deming and J.M. Juran, inspired the Capability Maturity Model. CMM divides software process improvement into five levels, in which each level represents a degree of maturity that the software developing organization plans to achieve. To give more guidelines and directions to software process improvement, CMM offers fifteen activities that correspond to the course of actions that need to be taken in order to achieve different levels of maturity. These fifteen activities are clustered into four categories.

The first category, organization, addresses issues manifested through policies, resource allocation, management supervision, overall human resources training, and specified responsibilities. Next, project management, covers the traditional planning, tracking, control, and subcontracting. Process management, the third category, aims at improving organizational maturity, and establishing the infrastructure. Process management encompasses the steps from process definition, through execution, control, and feedback. Technology constitutes the fourth category, and concerns the use of technology and an environment facilitated by appropriate tools. At each level of the maturity model, the organization following the CMM path will add layers of expected actions and tasks to the respective categories.

While CMM was intended for software from the start, the ISO 9000 series has a long history predating CMM, and is rooted in a more general environment. Applied to the software development process since 1991, the ISO-9000-3 standard still contains a language not familiar to the domain of software development. ISO-9000-3 addresses process consistency and control, and has little to say about the quality of the final products, customer satisfaction, business operating results, or bench marking. Additionally, ISO-9000 has no step-wise progression toward certification. While the standard assumes continuous improvement after certification, the certification could be perceived as the destination of the journey.

CMM has become the leading source in the US software industry, while globally, ISO-9000 is considered the recognized set of standards for software development. Sletzer et al. consider the ISO 9000 family as one of the most important standards in the field of software quality management in Europe.

Both CMM and ISO 9000 have their advocates and opponents. For example, Mark Schaeffer contends that "CMM has contributed to widespread success in assisting organizations to improve their efficiency in developing quality software product." Likewise, Gardner (asserts that the intent of the ISO 9000 series requirements is that a basic system be implemented to ensure customers that suppliers have the capabilities and systems to provide quality products and/or services. He adds that "The ISO 9000 series concentrate almost exclusively on results criteria..." Despite these sentiments regarding both models, practitioners have been slow to embrace SPI practices. Ritta Hadden notes that, "Many developers frequently do not see the benefits of managing requirements, nor do they feel the need to take time for project planning, tracking, and oversight, as advocated by the CMM." Pressman states that, "practitioners often perceive a disciplined process as a bottleneck to rapid progress. Many secretly view quality assurance activities as time-consuming, rather than time saving."

Sweeney and Bustard argue that the difficulty with both of these models is that they imply large incremental changes. With ISO requirements, a single level of attainment is defined, which can be achieved through substantial efforts and significant changes to existing working practices. With CMM, multiple levels of attainment have been defined, but again each step poses a major challenge and becomes progressively more difficult at higher levels.

In 1993, Johnson and Brodman embarked on a six months study of 35 organizations from government and industry. They reported that "most study participants who are seriously committed to software improvement realized at least some schedule, cost, and/or quality improvement." Since this study, there has been very little empirical data to provide some insight of how practitioners evaluate SPI initiatives and there has been no information to indicate a viable link between SPI and software product quality.

The major dispute facing the advocates of Software Process Improvement is whether there is enough evidence to show that SPI does indeed improve the quality of the final product and customer satisfaction. Also, there are conflicting opinions about whether SPI

methodologies are only appropriate for large software projects; success stories are based on large companies, which have sufficient resources to accommodate SPI requirements. The reality of the 21st century is that there are many small companies developing small-to medium-size products, which, either as a stand-alone product or embedded in a larger system, are as important as their larger counterparts.

The purpose of this paper is to get an empirical perspective on the deployment of SPI techniques and their impact on product quality and productivity. We compared and contrasted responses of SPI users and non-users to identify those factors that could possibly influence the adoption of a software process improvement technique. The following section describes our survey, and in section three we present the results and analyze them in the context of the management of information technology. Section four offers a conclusion and a summary of our findings.

## THE SURVEY

We chose four professional organizations in New England and distributed our survey to the members of these four organizations: Software Process Improvement Network (SPIN), the Association of Computing Machinery (ACM) Boston Chapter, the American Society for Quality (ASQ) of Massachusetts, and the New England Software Quality Assurance Forum (NESQAF). As members of these organizations, we had been regularly attending their monthly meetings. During the fall and spring of 1998, we contacted the chairpersons of these organizations, discussed the nature of the study, and received permission to conduct our survey. The method of data gathering was that of a semi-structured interview: we handed out the questionnaires to software developers and answered their questions as they were raised. Those members who were not part of a developing team did not participate, and although some developers worked for the same organization, almost all worked on different projects. Since we were only interested in the software developers' view of the product development process, we did not ask questions regarding organization type, size, or management. Our findings and analysis are based on the sixty-seven responses we collected.

The first part of the questionnaire was designed to identify the users and non-users of SPI methodologies, and to get a sense of the size and the type of software projects developed using SPI techniques. In our survey, we asked the respondents whether their organizations deployed ISO-9000, CMM (SEI), an in-house methodology, or no methodology at all. We also asked them to identify the business mode of their software development projects. The choices included in-house software development, contract for specific customer, retail for general purposes, and/or shrink-wrap. To determine the size of software developing projects, we asked for the size of developing team, and the range of time between the start and finish (first shipment) of the software development project.

In the second part of the survey, we asked questions pertaining to the impact of SPI methodologies on cost, schedule, and quality. We asked whether using an SPI methodology had significantly increased or decreased the actual cost versus scheduled cost. We also inquired about whether using SPI had significantly increased or decreased the actual delivery date versus the scheduled date. Questions regarding quality were in two parts: first, the impact of SPI on customer satisfaction; second, the impact of SPI on the quality of the product. While those who were using SPI could report on these aspects based on their experience, others could only express their opinion based on their expectation and what they perceived the impact would be.

The quality of a software product depends on good design, assuring the software product's correctness, maintainability, and verifiability. Another aspect of quality is performance, which depends on the software product's efficiency, integrity, reliability, usability, and testability. Finally, a high quality software product should conform to factors affecting its adaptation. These factors are expandability, flexibility, portability, reusability, interoperability, and intra-operability. We adopted the traditional definitions for these quality factors, which is presented in the table below.

Table 1
Software Quality Factors

| |
|---|
| **Quality of Design** |
| **Correctness**.<br>Extent to which the software conforms to its specifications and conforms to its declared objectives. |
| **Maintainability**.<br>Ease of effort for locating and fixing a software failure within a specified time period. |
| **Verifiability.**<br>Ease of effort to verify software features and performance based on its stated objectives. |
| **Quality of Performance** |
| **Efficiency**.<br>Extent to which the software is able to do more with less system resources (hardware, operating system, communications, etc.). |
| **Integrity**.<br>Extent to which the software is able to withstand intrusion by unauthorized users or software within a specified time period. |
| **Reliability**.<br>Extent to which the software will perform (according to its stated objectives) within a specified time period. |
| **Usability**.<br>Relative ease of learning and the operation of the software. |
| **Testability**.<br>Ease of testing the program to verify that it performs a specified function. |
| **Quality of Adaptation** |
| **Expandability**.<br>Relative effort required to expand software capabilities and/or performance by enhancing current functions or by adding new functionality. |
| **Flexibility**.<br>Ease of effort for changing the software's mission, functions or data to meet changing needs and requirements. |
| **Portability**.<br>Ease of effort to transport software to another environment and/or platform. |
| **Reusability**.<br>Ease of effort to use the software (or its components) in other software systems and applications. |
| **Interoperability**.<br>Relative effort needed to couple the software on one platform to another product and/or another platform. |
| **Intra-operability**.<br>Effort required for communications between components in the same software system. |

**Source:** (11) The Handbook of Software Quality Assurance, Prentice Hall, 1998.

In the third part of the survey, we asked questions regarding the importance of quality factors to software developers. We asked the respondents to rank the impact of the software process improvement methodologies on their development projects, and the importance of the quality factor for their product and/or their enterprise. The ranking was from very low, low, average, high, to very high. We wanted to examine whether the importance of software quality factors could be a possible determinant for the deployment of software improvement methodologies.

## THE RESULTS

In what follows, we summarize our findings and analyze how they contribute to long-term managerial knowledge. To that end, we separate projects that were developed using an SPI methodology from those that were not, and compare them by their type and size to determine whether type and size were among the factors influencing the use of SPI methodologies. Table 1 depicts the percentage of software projects that have been developed using a SPI Methodology.

Table 1

| | ISO | CMM | Other |
|---|---|---|---|
| SPI | 28% | 21% | 22% |
| Non-SPI | 40% | | |

Note: 12% of organizations use both ISO and CMM.

Our survey revealed that 60% of interviewed companies use some SPI methodology, while 40% replied "none" to the question of which SPI methodologies are used by their organizations. Considering that adopting such methodologies is voluntary, one may assume that the 60% of those organizations employing them do so because they expect some quality improvements and/or cost reductions. Our findings do not support such assumption; rather, it appears that the appeal of SPI is based on the perception of quality it creates in the mind of the customer. Thus, the type of the project and its potential users may have an impact on the deployment of SPI.

**Project Type**: The project type may have some ramifications on the managerial decision made regarding the adoption of a SPI methodology. Table 2 summarizes our findings in this respect.

Table 2
% of type of software projects

| | In-house | Contact | General/retail | Shrink Wrap |
|---|---|---|---|---|
| SPI | 50% | 60% | 20% | 15% |
| Non-SPI | | | | |

Note: some companies develop two or three types of software

The most noteworthy finding regarding the type of projects developed using SPI was that almost two-thirds of them were developed on a contractual basis, whereas only one-third of the projects that did not use SPI were contractual. This makes sense because the ISO in particular is based upon the use of a contractual agreement, and one of the criteria, which determine whether suppliers get a contract, is their ability to deliver quality software. Therefore, it is imperative that suppliers have a system in place to demonstrate that they take serious measures for delivering quality products.

The percentage of software products developed for internal use, general retail, or shrink-wrap did not deviate considerably between SPI and non-SPI users. In the managerial context, these findings imply that the adoption of an SPI methodology can be used as a certificate for quality performance, and perhaps as the most important consideration for selecting software development outsourcing. While meeting standards is crucial for contractual software products, in-house software projects are only scrutinized by internal customers, and the image of using SPI does not add any aura. Retail software products, on the other hand, are often evaluated by the practitioner press in term of functionality, added value, and reliability. These competitive benchmarks are regularly published and become the most important competitive advantage.

Our findings validate the assertion that since the quality of in-house and retail software are measured after development by either the internal customer or the market analyst, there is less need for image building. The contractual software developer, on the other hand, needs to create an image of quality awareness that could be sustained using a SPI methodology.

**Project Size:** The size of the projects was derived from two factors: development team size, and the time between the start and finish (the first revenue shipment) of the software development project. For both factors, we asked for a range of values rather then a specific value, and used mean and standard deviation for lower and upper limits as an indication of size and its variability. Table 3 summarizes this information.

Table 3

Size of software projects

| | Avg. and SD for the size of developing team | | Avg. and SD for the time between start and the end of software development project in months | |
|---|---|---|---|---|
| | Lower limit | Upper Limit | Lower limit | Upper Limit |
| SPI | μ=5<br>Φ=8 | μ=31<br>Φ=47 | μ=6<br>Φ=7 | μ=20<br>Φ=11 |
| Non-SPI | μ=2<br>Φ=1.2 | μ=14<br>Φ=10 | μ=6<br>Φ=5 | μ=18<br>Φ=11 |

Note: the outliers for the size of development team for non-SPI have been removed

The range for average and standard deviation of the developing team for projects using SPI were larger than those for projects not using SPI. However, the same measures were much closer for development duration. Of interest is our interpretation that companies who adopt an SPI methodology apply it to small and large projects, and since time to market is still the most important criterion for the software development industry, more manpower is assigned to larger products to meet the deadline. Projects developed without using SPI seem to use smaller developing teams, although there were two very large projects with over 100 persons in a developing team, which we took out as the outliers. Based on these findings, our study provides partial evidence to support the observation made by some researchers that SPI is deployed for large projects, and mostly by large companies, who have the resources to bear the initial cost of undertaking a new technique.

**Impact on Cost, Schedule, and Quality:** Once we got an overall picture of the types and sizes of software development projects that deploy SPI, and compared them to those projects that do not use SPI, we wanted to compare these two categories on different grounds: productivity and quality. To that end, we asked the respondents to rate the impact of SPI on cost, schedule, and quality factors ranging from highly increased, increased, the same, decreased to highly decreased. We emphasize again, that while SPI users were expressing their opinion based on experience, the non-users were merely speculating what the possible impact of SPI on cost, schedule, and customer satisfaction would be if they had used it. For the sake of simplicity and clarity, we have combined highly increased with increased, and decreased with highly decreased impact. We discuss the contributions of our findings to managerial knowledge after all of the results regarding quality are analyzed. Tables 4 and 5 compare the responses from those who use SPI with those who do not.

Table 4
Impact on cost

| | Increased/Highly Increased | Same | Decreased /Highly Decreased |
|---|---|---|---|
| Impact of SPI on cost (actual cost vs. estimated cost) | 40% | 29.63% | 29.63% |
| Non SPI | 30% | 70% | 0% |

The percentage of SPI users are somewhat evenly divided over the level of impact of SPI on cost with a slight tilt toward increased cost. Most of the non-users believe that SPI has no impact on cost, and about one-third speculates that cost increases. The impact on scheduling showed similar results: more than 1/3 of SPI users believed it would increase, 1/3 said they anticipated no impact, and a little less than 1/4 reported a decrease in delivery time. About 2/3 of non-users speculated no impact, less than 1/3 believed increase, and only 10% felt that use of SPI methodology would decrease delivery time. Table 5 shows similar results for the impact of SPI techniques on scheduling.

Table 5
Impact on schedule

| | Increased/Highly Increased | Same | Decreased /Highly Decreased |
|---|---|---|---|
| Impact of SPI on Scheduling (actual delivery vs. estimated delivery) | 38% | 38% | 24% |
| Non SPI | 30% | 60% | 10% |

Based on these responses we may conclude that while some SPI users have experienced a decline in cost and delivery time, the majority does not perceive much reduction in cost or in scheduling.

**Quality factors:** To find out whether software developers perceive SPI as having an impact on these factors, and/or whether these factors are important enough to warrant the use of SPI, we asked questions regarding impact and importance. Respondents ranked the impact of SPI methodologies on quality factors from very low, low, average, high, to very high. We combined high and very high impact (HVH), and used this figure as the basis of our analysis. Table 6 depicts the percentage of respondents that perceive the impact of SPI on design quality factors as HVH.

Table 6
HVH Impact of SPI on Design Quality Factors

| Design Quality factors | SPI | non-SPI |
|---|---|---|
| Correctness | 48.6% | 33.3% |
| Maintainability | 38.9% | 40.0% |
| Verifiability | 47.2% | 40.0% |

Note: the percentage of SPI users and non-SPI users are based on different denominators, thus does not add up to 100%.

According to this table, the software developers who use SPI have a better perception about the impact of SPI techniques on correctness and verifiability than those who do not use SPI. Correctness and verifiability are emphasized by both ISO and CMM and are correlated, in that if a software product is verifiable, it has a better chance of being correct. The percentage of respondents that perceive the impact of SPI on maintainability as HVH is almost the same for both groups, and is not perceived as high as the other two factors by SPI users. This could be due to the fact the maintainability is not a focus point of either ISO or CMM. Altogether, the percentages of respondents who ranked the impact high or very high on each factor by either category are less than fifty percent; hence, we may conclude that deployment of SPI does not necessarily improve the quality of design.

Comparing the percentage of respondents of SPI users and non-users that perceive the impact of SPI on performance quality factors as HVH revealed interesting results. Table 7 depicts these results.

Table 7
HVH Impact of SPI on Performance Quality Factors

| Performance Quality Factors | SPI | Non-SPI |
|---|---|---|
| Efficiency | 25.7% | 23.1% |
| Integrity | 22.9% | 30.8% |
| Reliability | 56.8% | 30.8% |
| Usability | 43.2% | 41.7% |
| Testability | 55.6% | 23.1% |

While a much higher percentage of SPI users reported HVH Impact on reliability and testability, the percentages of both groups that viewed the impact of SPI on other performance factors as HVH were almost equal. Again, reliability and testability are addressed explicitly by ISO and CMM, and are correlated factors of performance. A software product that can be tested with ease has a better chance of being reliable. HVH impact on efficiency, integrity, and usability did not differ significantly between the SPI users and non-users. While a smaller percentage of both groups perceived the impact on efficiency and integrity as significant, more than 40% reported HVH impact on usability. Both ISO and CMM address efficiency and integrity only implicitly. We may conclude that the impact of SPI on product performance is varied and deployment of SPI does not have a concrete impact on product performance.

The difference was more significant for adaptation quality factors. Table 8 summarizes these results.

Table 8
The Impact of SPI on Adaptation Quality Factors

| Adaptation Quality Factors | SPI | Non-SPI |
|---|---|---|
| Expandability | 54.5% | 27.3% |
| Flexibility | 48.9% | 27.3% |
| Portability | 40.9% | 27.3% |
| Reusability | 46.9% | 36.4% |
| Interoperability | 31.3% | 27.3% |
| Intra-operability | 52.9% | 27.3% |

The percentage of SPI users who reported HVH impact on adaptation factors was well above that of non-SPI users (except for Interoperability, which shows only a slight increase). Both CMM and ISO recognize the importance of adaptation factors. In the dynamic software market where change management is an important criterion for success, adaptation factors such as expandability, flexibility, and reusability are crucial for the long run maintenance of a family of software products. Yet, our respondents show a mixed reaction to the impact of SPI on these factors, indicating inconclusive results.

**Customer Satisfaction**: The percentage of responses that reported HVH impact on customer satisfaction was quite significant for both groups. More than two-thirds of the SPI users reported a high to very high impact on customer satisfaction, and more than half of non-users speculated high to very high impact. Table 9 shows these results.

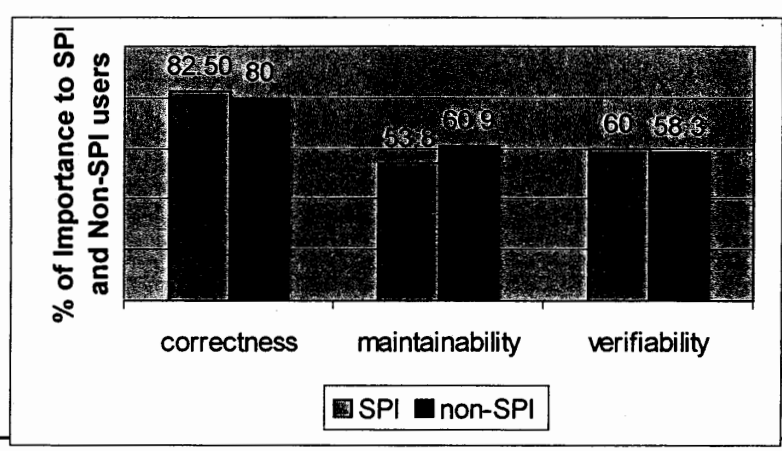
Table 9
Impact of SPI Methodologies on Customer Satisfaction

| | Increased/ Highly Increased | Same | Decreased/Highly Decreased |
|---|---|---|---|
| Impact of SPI on Customer Satisfaction | 71% | 18% | 11% |
| Non-SPI | 56% | 44% | 0% |

The majority of SPI users and non-users believed that the use of SPI would improve customers' perception of software quality. Table 4-9 summarizes the findings related to the quality aspect of this study, indicating that while SPI does not necessarily contribute to the reduction of cost and delivery time, it creates a perception of quality in the mind of customers. As for quality factors, at best the respondents showed a mixed reaction to the impact of SPI on them, leaving us to believe that SPI does not visibly improve the quality of design, performance, and adaption. However, the significant impact on customer satisfaction could be based on the mere perception that if an organization deploys SPI, it should improve the quality of products. Traditional literature refers to this phenomenon as perceived quality based on a subjective assessment resulting from an image. These findings raise a question among the software developing community about whether it is the image created by complying with some established guidelines, or the actual quality improvement that gains customer satisfaction and the market share.

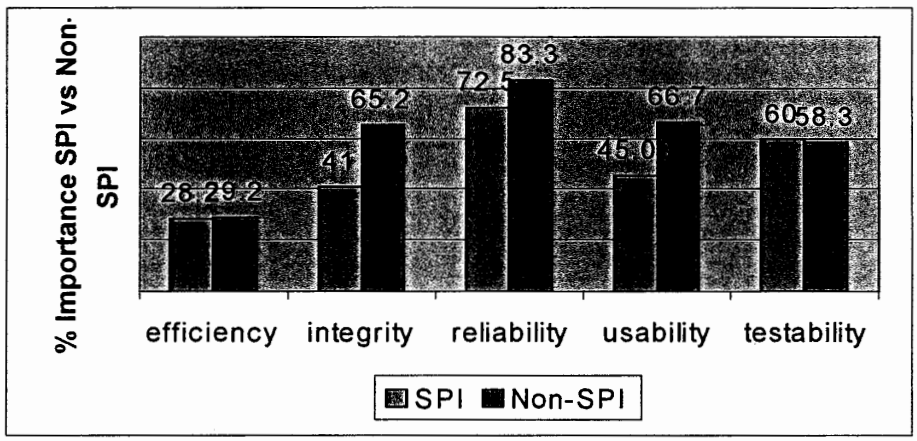
The last part of study focused on comparing the importance of quality factors between SPI users and non-users. Figure 1 compares the importance of design quality factors between the two groups.

Figure 1
Comparison of importance design factors



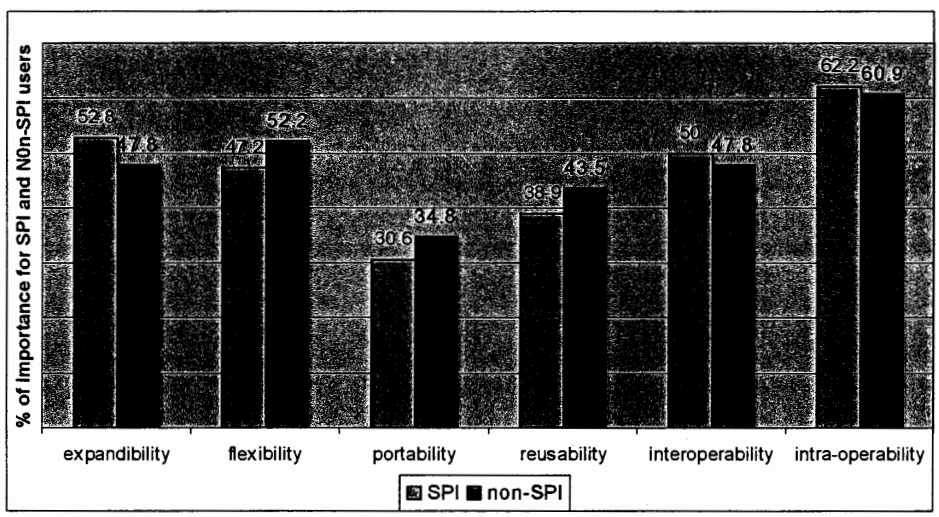
As the histogram indicates there is no significant difference between SPI users and non-users as they view the importance of design factors. Correctness and verifiability are viewed by a negligible percentage of SPI users as more important, and maintainability is perceived by a small percentage of non-users as more important. The importance of performance factors as it was viewed by users and non-users was more varied. Figure 2 depicts these differences.

Figure 2
Comparison of importance of performance factors




According to this histogram, non-SPI users consider integrity, reliability, and usability more important than SPI users, while both groups consider efficiency and testability equally important. There were also some differences between the two groups as they viewed adaptation factors. Figure 3 displays these differences.

Figure 3
The importance of adaptation factors



According to this histogram, while SPI users consider some factors of adaptation slightly more important, non-users see other factors as slightly more important. The difference, however, does not seem to be significant.

Since the above findings were somewhat inconclusive, we performed the Mann-Whitney-Wilcoxon test to determine whether there were significant differences between SPI users and non-users as they viewed the importance of quality factors. The Mann-Whitney-Wilcoxon test is a non-parametric alternative to the student's t-test for two mutually independent samples. Its less restrictive assumptions make this test appropriate for a wide variety of practical research situations in which classical statistics cannot be applied. The Mann-Whitney-Wilcoxon test requires only the assumption of any continuous distribution, no matter what shape, and data measured on an ordinal scale. Since our data met these requirements, we performed hypothesis testing and find the MINITAB solution with P= .4199, which fails to reject the null hypothesis of no significance difference. Thus, we may conclude that based on our collected data, the appreciation for quality factors by the developers are not the determinant of deployment of SPI methodologies.

## CONCLUSION AND SUMMARY

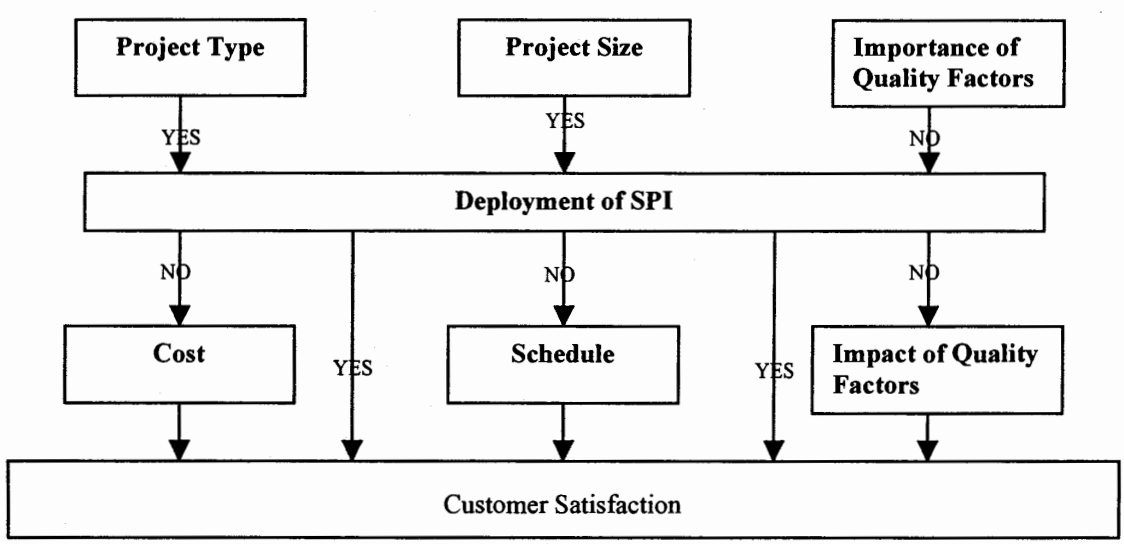
The contribution of this study to the management of information technology is twofold, based on both our approach and our findings. Unlike other studies on SPI that focus on the management viewpoint and organizational structure, this research is primarily focused on the *developers'* viewpoint on the deployment of SPI, specifically in New England region.

While exploratory in nature, this study provided some interesting insights to the deployment of SPI, the most noteworthy of which is identifying the factors that influence the deployment of software process improvement, and determining the actual and perceived impact of the deployment of SPI on quality and productivity. The diagram below depicts a causal model showing that while project type and project size have an impact on the deployment of SPI, the importance of quality factors to software developers does not influence the use SPI. Our survey results disclosed that projects developed for customers under contract are the most likely targets for deploying SPI methodologies. The results also indicated that SPI methodologies, once adopted, are deployed for large and small projects alike.

Also, the diagram shows that based on developers' opinions, we may conclude that although using SPI does not necessarily produce a quality software product at a reduced cost or delivery time, it creates a perception of quality, which leads to the customer satisfaction.

There was no significant difference between the percentage of SPI users and non-SPI users in the way they perceived the importance of software quality factors. Thus, we may conclude that based on this study the importance of quality factors is not a determinant factor on the adoption of the methodologies.

Causal Model for SPI Deployment



Our study is descriptive in nature and has limitations regarding the sample. There might be some biases from respondents (especially SPIN members) towards the impact of SPI methodologies on quality, timeliness, and reduced cost. This bias may have been exhibited on the impact on some very important quality factors such as correctness and reliability, which were perceived much more favorably by SPI users than those who did not use SPI.

Since we could not be sure that whether this sample could be a true representation of software developers in New England, we did not perform any inferential statistics that require scientific sampling procedure. We did use the Mann-Whitney-Wilcoxon test (a non-parametric approach, which does not require strict assumptions or a sampling method) to compare the importance of the quality factors between SPI users and non-users. For future research, we need to extend our sample to a much broader base using a sampling method that enables us to make inferences about a target population.

## REFERENCES


1- Cooper, J.D., and M.J. Fisher (ed.) *Software Quality Management*, Petrocelli Book, New York, 1979.

2- Fayad, M.E., and Laitinen, M., "Process Assessment Considered Wasteful" *Communications of the ACM*, November, Volume 40, Number 11, 1997, pp. 125-128.

3- Gardner, B.R., "ISO 9000 & TQM: Is It OK to Choose?" *The Community Quality Journal*, 1997.

4- Gray E.M. and Smith, W.L. " On the limitation of Software Process Assessment and the Recognition of a Required Re-Orientation for Global Process Improvement," *Software Quality Journal*, Volume 7, 1998, pp. 21-34.

5- Hadden, R., "How Scalable are CMM Key Practices," *Crosstalk: The Journal of Defense Software Engineering*, April 98, pp. 18-23.

6- Humphrey W.S., *Managing the Software Process SEI Series on Software Engineering*, Addison Wesley, 1989.

7- Johnson, D.I. and Brodman, J.G., "Realities and Rewards of Software Process Improvement," *IEEE Software*, Volume 13, Number 6, 1996, pp. 99-101.

8- Kuvaja P., "Improving Embedded Systems Quality With PROFES Methodology," Tutorial at PROFES '99 Conference, Oulu, Finland, June 22nd, 1999.

9- Pressman, R., "Software Process Perceptions," *IEEE Software*, Volume 13, Number 16, November 1996, pp. 16-18

10- Schaeffer, M.D., "Capability Maturity Model Process Improvement," *Crosstalk: The Journal of Defense Software Engineering*, May 1998, Volume 11, Number 5.

11- Schulmeyer, G. and McManus, J.I. *The Handbook of Software Quality Assurance*, Prentice Hall, Upper Saddle River, NJ. 1998.

12- Statz, J., Oxley, D., and O'Tool, P., "Identifying and Managing the Risks of Software Process Improvement," *Crosstalk: The Journal of Defense Software Engineering*, Volume 10, Number 4, April 1997.

13- Stelzer, D., Mells, W., and Herzwurm, G., "A Critical Look at ISO 9000 for Software Quality management." *Software Quality Journal*, Volume 6, 1997, pp. 65-79.

14- Sweeney, A., and Bustard, D.W. "Software Process Improvement: Making it Happen in Practice," *Software Quality Journal*, Volume 6, Number 4, 1997, pp.265-274.


## AUTHORS' BIOGRAPHIES

***Noushin Ashrafi*** *is an Associate Professor in the Management Science and Information Systems Department, College of Management at the University of Massachusetts Boston. Dr. Ashrafi received her Ph.D. in Management Sciences and Information Systems from the University of Texas at Arlington. Her current areas of research include software reliability, fault-tolerant software, applications of loglinear modeling, and the utilization of bayesian methods for estimating and predicting software reliability. Dr. Ashrafi has published*

*several papers on these topics and delivered presentations at numerous conferences.*

***Jean-Pierre Kuilboer** is an Assistant Professor in the Department of Management Science and Information Systems at the University of Massachusetts Boston. Dr. Kuilboer received his Ph.D. in Information Systems from the University of Texas at Arlington. He has been an active member of professional organizations such as the IEEE, ACM, AIS, and SPIN (Software Process Improvement) group. His Information Systems publications have appeared in refereed journals such as*: Information & Management, Information and Software Technology, the Journal of Accounting and Computers, the Journal of Database Management, Information Systems Management, Data Base, Journal of Information Systems Education, Computers and Society, *and* Data & Computer Communications.