Information Systems Frontiers 1:3, 267–277 (1999)

Software Process Models and Project Performance

M.S. Krishnan
University of Michigan Business School, Ann Arbor, MI 48109
E-mail: mkrish@umich.edu

Tridas Mukhopadhyay
Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213
E-mail: tridas@cmu.edu

Dave Zubrow
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213
E-mail: dz@sei.cmu.edu

Abstract. **In this paper we review the progress in software process research and the role of process improvement in enhancing business outcomes of software projects. We first describe the process view of software development. Next, we review the literature on software process research and discuss some of the leading software process models. The business value of software process improvements and empirical evidence from the software industry are also discussed in this paper. We conclude with a discussion of current challenges in software process research and directions for future research.**

Key Words. **software process models, CMM, software project performance**

1. Introduction

As computers evolved from special dedicated machines to ubiquitous entities in our daily life, the reliance of our modern society on computer software has grown significantly in the last few decades. Computer software is embedded in products ranging from automobiles and refrigerators to cardiac pacemakers. As a consequence, it is imperative for software products to function as per specifications. Indeed, the impact of software quality problems can be very serious. For example, problems with ambulance scheduling software in the United Kingdom resulted in the loss of a life due to delay in the arrival of ambulance (Flowers, 1996). Computer software also plays a critical role in determining the success of businesses and government in our society. Hence the development and maintenance of software have emerged as major concerns for the software industry in the last few decades.

In spite of the fact that the use and management of software is almost a necessity for the success of businesses and government, senior executives in most corporations are still troubled by chronic problems in software development and maintenance. These problems are usually associated with schedule delays, budget overruns, unacceptable product quality and dissatisfied customers. Such problems were together referred to as ''software crisis'' in the late 1960s and still continue to plague software organizations. For instance, a report by SRI International found that ''fewer than 1% of commercial software projects finish on time, on budget, and to specification'' (Dewey, 1988, p. 2). More recently, a 1995 survey conducted by the Standish Group (Johnson, 1995) reported that 31% of IT projects are cancelled prior to completion. Furthermore, the survey reported that nearly 53% of projects will incur significant (nearly 200%) overruns compared to initial budgets, and often contain reduced functionality compared to original requirements. A recent American Management Association survey of 400 CEOs indicated that only 55.6% of their information systems technology

projects met the initial budget targets and only 52.2% met the initial schedule targets (AMA Newsletter for Chief Executives, 1998). Additionally, software problems now receive widespread attention such as the baggage handling software which delayed the opening of the Denver Airport (Gibbs, 1994).

The business impact of the above problems in the software industry is significant and often threatens the profitability and survival of firms. The enormous growth in the demand for software has fueled the increase in the number of firms entering the industry and has intensified competition. In order to succeed in the market, software managers in software firms are striving for lower cost, on time delivery, higher quality and customer satisfaction. As a consequence, these endemic problems in software development have caught the attention of researchers both in the academia and industry, and a number of possible explanations and solutions have been proposed.

The proposed solutions to the software problems can be classified into models, methodologies, and tools. Organizations adopt these solutions to increase software development efficiency and quality, and facilitate better control of the software development activities. The waterfall model for software development and the structured analysis and design techniques introduced two decades ago provided a framework to represent in diagrams, the various phases of product development and the specific interactions among various modules in a software respectively (Yourdon, 1991; Boehm, 1981). This structured approach to business development was a step forward in improving program understanding and controlling software complexity.

A large number of CASE (Computer Aided Software Engineering) tools were introduced in the later half of 1980s to improve the efficiency and quality of software development. These tools were invented under the premise that many activities in software development are mechanical, and can be automated to a large extent. Hence the objective of these CASE tools was to provide efficiency and better control through automation of activities in various phases of software development as well as the interactions among these phases. However, empirical research on software projects does not provide a clear evidence of success in support of the use of CASE tools in software projects. Rather a number of studies have identified the importance of effective management of various activities in software development and the role of management innovations in the success of these technological solutions (Dewey, 1988; Martin and McClure, 1988).

As a major step towards addressing the problems in software industry, a new initiative with a focus on the software development process emerged in the late eighties. The Software Engineering Institute at the Carnegie Mellon University began its work on the Capability Maturity Model® (Capability Maturity Model and CMM are registered with the U.S. Patent and Trademark Office) to aid the Department of Defense in the acquisition of software intensive systems (Humphrey, 1989). Built on broader theories of quality and continuous process improvement, but tailored to software development, the CMM® provided the DoD a standard means for measuring the contractor capability via its definition of process maturity. As the CMM was widely adopted as a standard process model in the defense sector, commercial organizations within these corporations began to investigate whether they could benefit from the approach to software process improvement outlined in the CMM. Momentum grew over the years to today where software process improvement has become a global movement. In the last few years software process improvement has emerged as an integrated solution to software problems in various corporations. In addition, empirical evidence of the benefits of process improvements are now widely recognized (Herbsleb et al., 1997; Krishnan, 1996; Gopal, Mukhopadhyay and Krishnan, 1999).

In this paper we review the progress in software process research and the role of process improvement in enhancing business outcomes of software projects. We describe the process view of software development in section two. In section three we review the literature on software process research and discuss some of the leading software process models. Business value of software process improvements and empirical evidence from the software industry are discussed in section four. We conclude in section five with a discussion of current challenges in software process research and directions for future research.

2. Process View

A system of operations or steps in producing a product or service is termed a process. Software development

has been traditionally viewed as a set of purely engineering tasks. In contrast, a process view of software development presents the set of tasks involved in software development and maintenance as an integrated process that can be controlled, measured and improved (Humphrey, 1989). This process focus in software originated from analogies drawn by viewing software development in comparison to other mature fields such as manufacturing.

In successful manufacturing firms, all tasks and roles involved in the shop floor right from procurement of raw materials to shipment of the finished product are well defined as a process and adequate checks are in place to ensure process control. For example, the finished products are checked for quality within well defined tolerance levels, and in the event of any deviation, the source of defects in the process are identified and adequate measures to correct and improve the process are taken. As a consequence, product quality, development time and cost are relatively more predictable and controlled in the manufacturing industry as compared to the software industry. Hence the primary motivation for a process view of software development is the belief that the problems referred to as ''software crisis'' earlier can be solved to a large extent by incorporating all the tasks involved in software development into a well defined process and subsequently aligning the incentive structures within the software organization to ensure the following: (a) the defined process is followed; (b) any deviations from the process are tracked; (c) process outcome metrics are tracked and used in guiding periodic improvements to the process.

The first initiative on this process view of software development was pioneered in the late 1980s by a group headed by Watts Humphrey at the Software Engineering Institute (SEI) of Carnegie Mellon University (Humphrey, 1989). This subsequently led to well defined process models and frameworks that facilitate software organizations to define their software process, ensure strict adherence to the process standards, track the performance of their projects and improve their existing processes (Paulk et al., 1994; El Emam et al., 1997; Kuvaja et al., 1994). It is argued that a process based approach to software development induces discipline. It is believed that this discipline will subsequently manifest itself into a consistent pattern of behavior by individuals and groups of people following a common process. Since a defined process will describe how to act or react to certain situations that may arise in software development, a process based approach to software development will facilitate repeatability and a systematic data driven solution to problems faced by software managers.

The benefits of this process view are apparent specifically in large software projects involving many software professionals. In large software projects with multiple groups, it is important that the priorities, behaviors, and actions of various individuals and groups are aligned with the common goal of the project. A process view of software development provides a unified view and the needed alignment of the entire set of tasks and actions across various groups and individuals in the project. In the absence of such a unified view towards a common set of goals, the behavior and actions of various groups may contradict each other and subsequently affect the performance of the project. In summary, a process approach to software development brings discipline to various tasks in software development and this discipline leads to consistency and uniformity in products delivered at the end of these tasks. Instilling such discipline in all tasks in software development and maintenance will lead to an increased capability of the process and better quality, schedule and cost results in software projects.

3. *Software Process Models*

Software process research in the last decade has led to several approaches for software organizations to use in characterizing the state of their current software practices in consistent terms, and in setting goals and priorities for improving their processes. In particular, a number of models and frameworks for assessing software process capability of organizations have been developed. Some of these frameworks for software process improvement are SEI CMM, SPICE, ISO-9000-3, Bootstrap, and Trillium (Emam and Goldenson, 1996; Paulk, 1995; Paulk et al., 1993; Kuvaja et al., 1994). Most of these frameworks are based on fundamental principles of quality management. The basic premise of these frameworks is that consistent application of well-defined and measured software processes, coupled with continuous improvement of process, will substantially improve the productivity of software organizations and the quality

of their products. We next briefly discuss some of the popular software process models and frameworks for process capability evaluation and assessment.

3.1. Capability Maturity Model

The Capability Maturity Model (CMM) was developed by the Software Engineering Institute (SEI) of Carnegie Mellon University. As depicted in Table 1, the CMM specifies five maturity levels to assess an organization's process capability by measuring the degree to which processes are defined and managed. These levels range from no process or ad hoc rules in software development at the lowest maturity level to continuously improving and optimized process at the highest level. Each maturity level consists of several key process areas (Paulk et al., 1993, 1994) (see Table 1). Key process areas are classified into goals and common features. In each key process area, several goals are defined to represent the desired outcomes of the process. Guidance for realizing the goals is provided in the form of key practices grouped in terms of practices for implementing and institutionalizing the process. A series of such practices and process improvement initiatives lead to the transition of an organization's process to the next higher maturity level.

3.2. ISO-9000

ISO-9000 is a general quality standard from the International Standards Organization. This quality standard was originally designed for the manufacturing industry and was later extended to service and software industries. Several firms have been designated by the International Standards Organization as ISO-9000 auditors. These firms then certify other organizations as ISO-9000 compliant based on a rigorous audit of quality practices followed by the organization in their internal processes. ISO-9000 is a family of standards and contains several parts. The set of standards stress the following three broad quality concepts (ISO, 1987):

i. An organization should achieve and sustain the quality of product or service produced so as to meet continually the customer's stated or implied needs.
ii. An organization should track outcomes of quality practices and provide confidence to its own management that the intended quality is being achieved and sustained.
iii. An organization should provide confidence to the purchaser that the intended quality is being, or will be, achieved in the delivered product or service provided. When contractually required, this provision of confidence may involve agreed demonstration of requirements.

Only some of the parts of the full ISO-9000 set of standards are relevant to software development. The part most relevant to software development is ISO-9000-3 (ISO, 1991; 1994). This part provides detailed quality guidelines for development, supply and maintenance of software products and services and includes an interpretation of the general ISO-9000 standard for a software organization. It includes guidelines for quality practices in both development and support activities. The specific activities addressed in ISO-9000-3 are listed in Table 2. Although ISO-9000-3 primarily addresses custom software development where specific software is

Table 1. *Overview of the SEI-CMM*

Level	Focus	Key process areas	
5 Optimizing	Continual process improvement	Defect prevention, Technology change management, Process change management	Quality Productivity
4 Quantitatively Managed	Product and process quality	Quantitative process management, Software quality management	
3 Defined	Engineering processes and organizational support	Organization process focus, Organization process definition, Training program, Integrated software management, Software product engineering, Intergroup coordination Peer reviews	
2 Repeatable	Project management processes	Requirements management, Software project planning, Software project tracking & oversight, Software subcontract management, Software quality assurance, Software configuration management	Risk
1 Initial	Competent people and heroics		Waste

This table is a courtesy of Mark C. Paulk of the Software Engineering Institute at Carnegie Mellon University.

Table 2. *Activities in software process addressed by ISO-9000 quality system*

Lifecycle activities	Support activities in software development
Contract review	Configuration management
Purchaser's requirement specification	Document control
Development planning	Quality records maintenance
Quality planning	Measurement and metrics
Design and implementation	Rules, practices and conventions
Testing and validation	Tools and techniques
Product acceptance	Purchasing
Replication, delivery and installation	Training
Maintenance	

developed as per the purchaser's requirements, these quality concepts can also be used in projects that develop packaged software for a market.

Note that in spite of the fact that both ISO-9000 and the CMM model for software process improvement address the importance of a defined process with guidelines and practices to achieve quality, there are some inherent differences between the two approaches (Paulk, 1995). Unlike the CMM model discussed earlier that provides a framework for continuous process improvement, the ISO-9000-3 aims primarily to establish an acceptable baseline for software process. The ISO standards address the minimum criteria for an acceptable quality system. In addition, the CMM can be used both for an internal assessment by the organizations and external audit, whereas ISO-9000 certification can be provided only by external auditors. Paulk (1995) provides a detailed analysis comparing and contrasting the CMM and ISO-9000-3 standards. Based on the ISO-9000 standards, the British Standards Institute (BSI) in collaboration with the UK Department of Trade and Industry (DTI) and software industry representatives have developed quality guidelines for software purchasers, software suppliers and quality auditors. This guideline is referred to as ''TickIT'' guide to Software Quality Management System Construction and Certification using ISO-9001.

3.3. SPICE

Software Process Improvement and Capability dEtermination (SPICE), aka ISO-15504, is an emerging international standard for conducting software process assessment. It provides a framework and set of requirements that can accommodate a variety of models of software practice and assessment methods (Rout and Simms, 1998). A set of requirements is specified that allow users to determine if their assessment conforms to the standard. This approach allows users of other process improvement models such as the CMM to comply with the standard without abandoning their existing improvement approach.

The SPICE document set, however, contains much more than the requirements. To support the standard and give organizations that are embarking on software process improvement a starting point, SPICE also includes a reference model of practices and an assessment method. The SPICE reference model follows a continuous architecture for capability. Thus each process category is rated separately on a scale from 0 to 5. This approach differs from a staged model like the CMM where rating is based on the implementation of clusters of process areas and practices and is reported for the organization as a whole.

SPICE is also seeking to harmonize with ISO-12207, the standard for Software Life Cycle Processes. Both standards address software development processes and hence, the need for harmonization of the standards. However, the SPICE reference model provides greater detail and serves as a suitable basis for conducting assessment and guiding improvement.

3.4. Trillium

The Trillium process model is based on the CMM and quality practices outlined in other models. A consortium of telecommunication companies headed by Bell Canada developed this model (Trillium, 1994). A unique feature of this model is that it was designed to primarily address the specific requirements of software development in the telecommunications industry. This model identifies the key industry practices and provides guidelines for

improving the software development process capability in a competitive business environment. The focus is primarily on delivering software products that meet the expectations of the customer and exhibits highest quality, fast development schedule and lower lifecycle costs (Trillium, 1994).

Similar to the five levels of process maturity in the CMM, the Trillium model also specifies five levels of process improvements. As shown in Table 3 these levels range from unstructured, ad hoc process at the lowest level to fully integrated process at the highest level where formal methodologies are extensively used, repositories for development history and process are effectively utilized and process improvement is engineer-driven. The process risk gradually decreases from ''High'' to ''Lowest'' with the increase along the five levels. The Trillium model contains eight capability areas and each capability area contains several ''roadmaps'' with practices at multiple process improvement levels as shown in Table 4. For a given roadmap, the level of practices is based on their respective degree of maturity. The fundamental practices are at the lower levels, whereas more advanced ones are at the higher levels. In order to increase effectiveness of higher level practices, it is recommended that the lower level practices be implemented and sustained (Zahrin, 1998).

The Trillium model is quite similar to the CMM. The key process areas of the CMM have been condensed into fewer capability areas that are elaborated and expanded into various roadmaps. In addition, Trillium also draws from business process reengineering and total quality management. The differences between Trillium and the CMM model are outlined by Popel and Wise (1996).

3.5. *BOOTSTRAP*

The BOOTSTRAP methodology for software process assessment and improvement originated in a European Community project (ESPRIT) with a focus on evaluating investments in technology (Kujava, 1994). The primary goal of this methodology was to aid adoption of software engineering technology in the small and medium sized organizations in the European software industry. This methodology addresses process management issues both at the organizational level and at the level of individual software producing units within the organization. Unlike some process assessment frameworks discussed earlier, the BOOTSTRAP methodology not only provides an assessment of the

Table 3. *Trillium process improvement levels*

Level	Description	Risk
Level 1: Unstructured	Hero-driven product development	High
Level 2: Repeatable and project oriented	Project management process in place	Medium
Level 3: Defined and process oriented	Process-oriented engineering standards in place	Low
Level 4: Managed and integrated	Manager-driven process improvement	Lower
Level 5: Fully integrated	Fully integrated	Lowest

Table 4. *Trillium capability areas and roadmaps*

Capability Areas	Roadmaps
Organizational process quality	Quality management, Business process engineering
Human resource development and management	Human resource development and management
Process	Process definition, Technology management, Process improvement and engineering, Measurements,
Management	Project management, Subcontract management, Customer-supplier relationship, Requirements management, Estimation
Quality systems	Quality system
Development practices	Development process, Development technique, Internal documentation, Verification and validation, Configuration management, Reuse, Reliability management
Development environment	Development environment
Customer support	Problem response analysis, Usability engineering, Lifecycle cost modeling, User documentation, Customer engineering, User training

current practices followed within an organization, but also provides specific tools and methods to analyze assessment results. This methodology also aids in transforming the results of process assessment into an action plan to improve process maturity. Hence it provides a structured guide to sustain process improvements.

The BOOTSTRAP process architecture rests on a triad of process categories in the areas of Organization, Methodology and Technology. These three major categories consist of several process areas. Each process area specifies several key practices and activities. The important process areas within each of the three categories of the triad are shown in Table 5. The organization category addresses the process management issues at the organizational level and captures information on the resources and profiles in various projects of the organization and on the organization as a whole. The issues addressed in the process areas of methodology and technology category are more confined to the specific software projects.

The questions in the BOOTSTRAP assessment questionnaire are also divided in the three main categories as depicted in Table 5. The BOOTSTRAP methodology uses the same five level process capability scale used by CMM (as shown in Table 1). However, there are some salient differences in the way assessment scores are computed in this approach. First, the process areas defined in BOOTSTRAP architecture are not confined to one single capability/maturity level as in the CMM but span across several levels similar to the Trillium process model. Second, in the BOOTSTRAP method, the final process assessment output is not expressed as a single aggregate number indicating the maturity level. The final output is a profile of key attributes. The method offers a scoring algorithm to be used by assessment teams in analyzing responses to the questionnaire. Each question is measured on a four-point scale and is translated into numeric equivalents for aggregation to average scores. This scoring algorithm ensures that all the key attributes are directly represented on the five maturity scales (Kujava, 1994). The BOOTSTRAP methodology also captures the concept of organization wide quality system specified in ISO-9000 standards.

Table 5. *Process areas in BOOTSTRAP architecture*

Process category	Process areas
Organization	☐ Management practices
	☐ Quality management
	☐ Resource management
Methodology	Product engineering
	☐ Requirements analysis
	☐ Definition
	☐ Detailed design and implementation
	☐ Testing and integration
	☐ Acceptance and transfer
	☐ Operational support and maintenance
	Process engineering
	☐ Process description
	☐ Process control
	☐ Process measurement
	Life-cycle independent (support) functions
	☐ Project management
	☐ Quality assurance
	☐ Risk management
Technology	☐ Technology management
	☐ Product engineering technology

4. Business Value from Software Process Improvements

As discussed earlier, the CMM and various other software process models provide guidelines and frameworks to adopt disciplined methods and practices to software development and maintenance. The basis for all these process models rests on the belief that following a defined process with disciplined methods and practices will lead to significant business benefits for software organizations. These benefits are in terms of cost reductions, and quality and cycle-time improvements in software projects. In spite of these beliefs in substantial benefits from process improvements and disciplined practices, software organizations have been rather slow in adopting such practices until the past few years. There are several reasons for this resistance to adopt these practices by software organizations.

First, some software developers and managers believe that enforcing a rigid discipline to software development through a defined process may reduce the freedom for developers and curtail their innovation. However this belief is not correct (Humphrey, 1996). Second, software organizations often view the substantial initial effort required to institutionalize a defined process and the ongoing effort to follow the process as overhead expenses. Finally, since it is often

difficult in software projects to isolate the cause of improvements in costs or quality to a specific practice, there has been a concern regarding the effectiveness of software process improvements and related practices. This validity concern is mainly due to the lack of controlled and rigorous empirical evidence to support the benefits claimed from process improvement and related practices.

Several studies have been conducted to assess benefits from process improvements. Most of these studies report evidence in support of business benefits from software process improvement. We briefly discuss the key results of some of these studies and summarize their findings. Empirical evidence reported in support of software process benefits can be broadly classified in to the following three categories:

1. Survey reports based on responses from managers and developers in software development firms.
2. Longitudinal case studies of a few projects conducted within a company.
3. Empirical studies that estimate the effects of process on costs, quality and cycle-time using statistical models on pooled data from several projects.

Although all three categories of evidence contribute to our understanding of software process benefits, these results must be viewed in the context of their unique strengths and limitations.

4.1. Survey studies

Based on over 138 survey responses from senior technical personnel, project managers and members of SEPG in several organizations, Herbsleb et al. (1997) report consistent perceptions of improved project performance along several dimensions as process maturity increased. Deephouse et al. (1993) in another survey based on responses from SEPG members of several organizations report positive association between project performance and key practices followed to improve software processes.

In a US Air Force sponsored research to assess the quantifiable benefits and ROI for software process improvement efforts based on SEI CMM, Brodman and Johnson (1995), report benefits such as increased productivity and reduced development schedule. Although these reports improve our understanding of the perceived or actual benefits from software processes, it must be noted that results based on survey reports are subjective and may not provide quantitative evaluation of benefits. In addition, some survey reports capture merely perceived benefits from software process instead of actual benefits.

4.2. Case studies

Based on data collected from multiple case studies in thirteen organizations, Herbsleb et. al. (1997) report that organizations engaged in CMM based software process improvements experienced productivity gains from 9 to 67%, reduction in product development time ranging from 15 to 23% and reduction in post release defects ranging from 10 to 94%. An eight year longitudinal study of software projects at Raytheon reports a 190% increase in productivity and a decrease in product trouble rates from 17.2 to 4.0 per thousand lines of delivered code (Haley, 1996). A six year study at Hewlett-Packard reports reduction in delivered defects from 1.0 per thousand lines of code to 0.1 per thousand lines of source code and cost savings of over $100 million through process improvements (Myers, 1994).

A large Italian software house has reported reduction in budget overruns in their projects through process improvement. A large utility company in Europe has experienced over 20% cost reduction in large projects by adopting a formal process for managing requirement specifications (Zahran, 1998). IBM Federal Systems Company that developed space shuttle flight software for NASA has initiated software process and quality practices to meet the highest reliability and safety standards required in these software systems. In two decades of process improvement, this organization has evolved the process capability to consistently predict its cost within 10% of actual expenditures, achieve three-fold increase in productivity with only one schedule overrun in fifteen years (Curtis and Statz, 1996). Reports on process improvement benefits from several organizations are summarized in Fox and Frakes (1997). A definite strength of case studies is the access to in-depth analysis based on detailed information from few projects. In addition, the benefits reported in case studies are based on objective data. However, there are also limitations in the case based evidence. First, the results are confined to a few projects and it may be difficult to statistically test the causality in business value benefits. Second, it may be possible that only results of successful projects are released.

4.3. Field studies

The evidence from field studies is based on data from a relatively large number of projects within a single organization or from several organizations. Krishnan et al. (1999) examine the effect of CMM KPAs on software quality and cost. Their analysis based on data from over 40 projects from a large software development laboratory in a Fortune 100 company indicates that consistently following CMM KPA practices improved quality, but did not show evidence of a direct effect on cost. Higher product quality, in turn, significantly reduced cost. A recent dissertation investigating the impact of software process maturity on software development productivity found that after normalizing for the effects of other effort influences, a one-level change in the rating of CMM process maturity level resulted in a 15 to 21% reduction in effort (Clark, 1997). This finding is significant since the analysis considered all the cost factors included in the COCOMO II cost estimation model (Boehm et al., 1995).

Gopal et al. (1999) study the effect of process factors on effort, cycle-time and rework in offshore development projects completed in India. Their analyzes of data from over 30 offshore software projects from a large software vendor show that software process factors significantly improve all three performance measures—effort, cycle-time and rework. In a different study, Harter, Krishnan and Slaughter (1998) investigate the relationship between process improvement, quality, cycle time, and effort for the development of thirty software products by a major IT firm. Their findings indicate that higher levels of process maturity as assessed by the CMM are associated with significantly higher product quality and higher cycle time and development effort. However, the reductions in cycle time and effort due to improved quality in their finding outweigh the increases from achieving higher levels of process maturity. Thus, the net effect of process improvement is reduced cycle time and development effort.

A primary limitation in the evidence reported in the above studies is that the data is often from one organization. To improve generalizability of these results, there is a need to pool data from several organizations to test the effect of process improvement on project performance. However, in combining data from several organizations, care must be taken to ensure that measures of productivity, quality, and other product metrics are uniform across organizations.

5. Conclusion

In this paper we have discussed the evolution of process based approach in software development and described various process models and frameworks in use. It is clear from the evidence discussed in section four that adopting disciplined practices through a defined process may provide significant return on investment. However, in order to sustain process improvement and prevent unwanted deviations from defined processes, process owners within organizations need to periodically present the business benefits of process adoption to its software community. Due to the intense pressure from the market and their customers to release products early, software managers commit a major mistake of viewing some parts of the defined process unnecessary. Unless this gets the attention of the group responsible for software process in the organization and the required changes to the process are implemented, this behavior of missing process steps becomes a norm. Hence correct monitoring of process execution especially for a few months after the process is in place is critical for successful adoption of software process. Senior management recognition and support for software process improvement programs is a must. As in any other change management exercise, the momentum required to sustain the major changes introduced through process improvement programs require the awareness and commitment from senior management.

Many challenges remain in this area. Although we have primarily discussed process definition and process improvement in this paper, for reaping business value benefits, software managers need to manage and improve people skills and use appropriate methodology and automated tools deployed in product development. For instance, rarely are process improvement activities undertaken in isolation. Organizations are dynamic and continually evolving. Developing knowledge of the relative contribution of process in light of changes in technology and human resources remains a fertile area for research. Likewise, little is known about how process improvement might vary in response to differing organizational goals and priorities.

In the packaged software industry, cycle time is often touted as the highest priority. In the contract software industry, cost may be the priority. In DoD and mission critical areas, quality, functionality and schedule may be preeminent. Finally, little is known

with respect to the comparative effectiveness of the various process improvement models. In particular, we need to compare process improvement efforts guided by a staged model like the CMM® with process improvement efforts guided by a continuous architecture like that embodied in SPICE. Of course, the answer is, it depends. To date, however, criteria or heuristics are lacking to help organizations make this choice. Empirical studies in the literature on process improvement benefits have primarily studied the CMM model. This is partly due to the early and widespread adoption of the CMM process model by commercial organizations. There is a need to study the cost benefit trade-offs of other software process models such as the SPICE or BOOTSTRAP. To further knowledge about these trade-offs, datasets based on the experience of multiple organizations need to be established. Furthermore, ways to gather comparable data or to normalize data from disparate sources need to be developed.

References

AMA *Newsletter for Chief Executives*. PRESIDENT, Spring 1998.

Boehm BW. *Software Engineering Economics*. New Jersey: Prentice-Hall, Inc., 1981.

Boehm BW, Clark B, Horowitz E, Westland C, Madachy R, Selby R. Cost models for future software life cycle processes: COCOMO2.0. *Annals of Software Engineering*. Arthur JD, Henry SM. eds. Amsterdam, The Netherlands: J.C. Baltzer AG, Science Publishers, 1995;57–94.

Broadman JG, Johnson DL. Return on investment (ROI) from software process improvement as measured by US industry. *Software Process* Pilot Issue, 1995;35–47.

Clark BK. The effects of software process maturity on software development effort, University of Southern California. *Doctoral Dissertation* 1997.

Curtis B, Statz J. Building the cost-benefit case for software process improvement. in: *Proceedings of 1996 SEPG Conference*. Atlantic City, New Jersey, May 20–23.

Deephouse C, Mukhopadhyay T, Goldenson DR, Kellner MI. Software processes and project performance. *Journal of Management Information Systems* 1996;12(3):185–203.

Dewey, Russell H. Software engineering technology and management. *SRI International, Report No. 762* 1988.

Dion, R. Process improvement and the corporate balance sheet. *IEEE Software*, 1993;10(4):28–35.

Dorling, A. SPICE: Software Process Improvement and Capability Determination. *Information and Software Technology* 1993;35:404–406.

El Emam K, Drouin, J-N, Melo, W. *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. IEEE Press 1997.

El Emam, Goldenson DR. Some Initial Results from the International SPICE Trials. *Software Process Newsletter, Technical Council on Software Engineering* IEEE Computer Society, 1996;6.

Flowers S. *Software Failure: Management Failure*. New York, NY John Wiley and Sons, 1996.

Fox C, Frakes W. The Quality Approach: Is it Delivering? *Communications of the ACM* 1997;40(6):25–29.

Gibbs W, Wayt. Software's chronic crisis. *Scientific American* 1994;86–95.

Gopal A, Mukhopadhyay T, Krishnan MS. The Role of Software Processes and Communication in Offshore Software Development. Forthcoming, *Communications of ACM* 1999.

Harter D, Krishnan MS, Slaughter S. The Effect of Process Improvement on Quality, Cycle time, and Effort in Software Product Development: A Field Study. *Working Paper, 1998, Graduate School of Industrial Administration*. Carnegie Mellon University, Pittsburgh, PA 15213.

Haley TJ. Software Process Improvement at Raytheon. *IEEE Software* 1996;13(6):33–41.

Herbsleb J, Zubrow D, Goldenson D, Hayes W, Paulk M. Software Quality and the Capability Maturity Model. *Communications of the ACM* 1997;40(6):30–40.

Humphrey WS. *Managing the Software Process*. Addison-Wesley, 1989.

Humphrey WS. *Managing Technical People: Innovation Teamwork and Software Process*. New York, NY Addison-Wesley, 1996.

Humphrey Watts Snyder Terry R, Willis Ronald R. Software Process Improvement at Hughes Aircraft. *IEEE Software* 1991;8(4):11–23.

ISO *Quality Systems: Model for Quality Assurance in Design/development, Production, Installation and Servicing*. ISO-9001, International Standards Organization, 1987.

ISO *Quality Management and Quality Assurance Standards, Part 3: Guidelines for the ISO 9001 to the Development, Supply and Maintenance of Software*. ISO-9000-3 International Standards Organization, 1991, 1994.

Johnson Jim. CHAOS: The dollar drain of *IT* project failures. *Application Development Trends* 1995;20(1):41–44.

Krishnan MS. Cost and Quality Considerations in Software Product Management. *Doctoral Dissertation*. Graduate School of Industrial Administration, Carnegie Mellon University, 1996.

Krishnan MS, Kriebel CH, Kekere S, Mukhopadhyay T. An Empirical Analysis of Quality and Productivity in Software Products. *GSIA Working Paper, 1999*. Carnegie Mellon University, Pittsburgh, PA 15213.

Kuvaja PJ, Simila L, Krzanik A, Bicego S, Soukkonen Koch G. *Software Process Assessment and Improvement: The BOOTSTRAP Approach*. UK: Blackwell, 1994.

Martin J, McClure CL. *Structured Techniques: The Basis for CASE*. New Jersey: Prentice-Hall, 1988.

Myers W. Hard data will lead managers to quality. *IEEE Software*, 1994;11(2):100–101.

Paulk MC, Weber CV, Garcia SM, Chrissis M, Bush M. Key Practices of the Capability Maturity Model, Version 1.1. *Technical Report, CMU/SEI-93-TR-25, 1993*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

Paulk MC, Weber C, Curtis B, Chrissis MB. *The Capability*

Maturity Model—Guidelines for Improving Software Process. Addison-Wesely, 1994.

Paulk MC. How ISO-9001 compares with the CMM. *IEEE Software*. 1995;74–83.

Popel B, Wise C. Trillium and the CMM: differences between the two models and assessment methods. *Proceedings of 1996 SEPG Conference*. Atlantic City, NJ, pp. 20–23.

Rout TP, Simms PG. Introduction to the SPICE Documents and Architecture. in *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*. El Emam K., Drouin, J-N, and Melo, W. eds. Brussels: IEEE Press, 1997.

Trillium: Model for Telecom Product Development and Support Process Capability, Release 3.0, *Technical Report*, Bell Canada Acquisitions, Canada, 1994.

Yourdon E. *Modern Structured Analysis*. Yourdon Computing Series, 1991.

Zahrin S. *Software Process Improvement: Practical Guidelines for Business Success*. New Jersey: Addison Wesley, 1998.

M. S. Krishnan is Assistant Professor of Computers and Information Systems at the University of Michigan. He holds a Ph.D. from Carnegie Mellon University. He won the Best Dissertation Award from the International Conference on Information Systems in 1997. His research primarily focuses on software development productivity. His research appears in *Management Science, Harvard Business Review IEEE Transactions on Software Engineering, Decision Support Systems, Information Technology and People* and *Communications of the ACM*.

Tridas Mukhopadhyay is Professor of Information Systems at Carnegie Mellon University. He holds a Ph.D. from the University of Michigan. He is the Director of the Master of Science in Electronic Commerce program at CMU. His research appears in *Information Systems Research, Communication of the ACM, Journal of Manufacturing and Operations Management, MIS Quarterly, Omega, IEEE Transactions on Software Engineering, Journal of Operations Management, Accounting Review, Management Science, Journal of Management Information Systems, Decision Support Systems, Journal of Experimental and Theoretical Artificial Intelligence, Journal of Organizational Computing, International Journal of Electronic Commerce, American Psychologist* and other publications. He is on the editorial board of *Information Systems Frontier, Journal of Management Information Systems, Journal of Organizational Computing, Management Information Systems Quarterly, Journal of the Association for Information Systems*, and *Information Systems Research*.

Dave Zubrow is Team Leader at the Software Engineering Measurement and Analysis group of the Software Engineering Institute at Carnegie Mellon University. He holds a Ph.D. from Carnegie Mellon University. He has researched extensively on software process management and measurement. He is a member of the editorial board for *Software Quality Professional*.