Organizational Management Practices for Achieving Software Process Improvement

Ronald Kirk Kandt

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California USA
ronald.k.kandt@jpl.nasa.gov

Abstract. The *crisis* in developing software has been known for over thirty years. Problems that existed in developing software in the early days of computing still exist today. These problems include the delivery of low-quality products, actual development costs that exceed expected development costs, and actual development time that exceeds expected development time. Several solutions have been offered to overcome our inability to deliver high-quality software, on-time and within budget. One of these solutions involves software process improvement. However, such efforts often fail because of organizational management issues. This paper discusses business practices that organizations should follow to improve their chances of initiating and sustaining successful software process improvement efforts.

1 Background

In the early 1970s, software engineering teams seldom produced applications on time or within budget. When completed, applications often did not meet system requirements or the expectations of their customers. Today, this problem still exists. For example, *Software Engineering Notes* publishes tens or hundreds of software failures in every issue, which demonstrates how widespread the problem still is.

To overcome these problems, the software industry has adopted several software process standards, requirements, and guidelines. However, overcoming these problems is an extremely challenging undertaking that is seldom undertaken properly since organizations largely ignore or incorrectly handle the human issues surrounding organizational change [9, 14, 35, 36, 39]. Consequently, fifty-five to ninety percent of all technology objectives fail because such issues were not properly addressed [28].

This paper discusses those practices that the author has found to significantly ease the introduction and sustainment of a software process improvement program. This information represents the collective wisdom of the author and several other people, gained through both success and failure.

2 Organizational Management Practices

To increase the success rate of software process improvement efforts, organizations need to perform several practices that reduce the number of social and cultural problems that occur when organizational change is introduced. Descriptions of some of these practices follow.

2.1 Responsibilities of the Organization

Define an organizational vision. A vision statement identifies what an organization should be, do, and create. Hence, a vision statement should identify the financial, human, and product values of an organization, reflect the organization's uniqueness, and set standards for productivity and quality that guides how work is done [18, 27]. Every vision statement should also define a core strategy based on achieving operational excellence, product leadership, or customer intimacy. Furthermore, a vision statement of an organization should be refined by lower-level organizational units and links should be created between each job description and these vision statements. This traceability permits each individual to understand his or her contribution to the organization and the relevance of proposed organizational changes. Without logically consistent, cascading vision statements people will tend to resist change because they will not understand the relevance of a proposed change to the organization or how it affects them.

The top-most software development organization must create a vision statement that identifies software process improvement as a goal of the organization and articulate a compelling need for it. A compelling need for software process improvement is a forceful, convincing, and necessary argument that identifies its usefulness or merit. Hence, compelling needs usually support the business needs and goals identified in higher-level vision statements [4, 13, 30, 32, 44]. Compelling needs encourage people to embrace change and increase the odds of a successful software process improvement effort [16, 37]. Unfortunately, the survival of an organization, and the jobs of its employees, is often the compelling need for change [9, 40]. Note that most organizations that have achieved a Capability Maturity Model Level 3 rating are Department of Defense contractors, who were coerced into achieving higher levels of maturity to bid on contracts and to receive preference on responses to government solicitations.

There are three common ways to identify a compelling need. First, an organization can compare itself against its competitors and other best-of-breed organizations. Second, it can identify and measure its core business needs and processes [17]. Third, it can identify business threats and opportunities [16]. Software process improvement efforts commonly use the first two methods and could benefit by using the third approach. Regardless, an organization should use only the most compelling threats and opportunities to convey an understanding of the reasons for change to its employees. Furthermore, an organization should communicate the compelling need of a change to its people in the terms most important to them [27]. Thus, a communication strategy

should relate a software process improvement effort to the roles that people perform in satisfying the organizational vision.

Commit to a software process improvement program. Achieving a commitment to change requires one or more key individuals to sponsor it and gain the commitment of line managers and other leaders [8, 20, 32]. A change sponsor must have the authority to legitimize a change, provide adequate resources to it [23], and ensure that it becomes an organizational goal. In addition, change sponsors must be willing to publicly support the change, demonstrate behavior consistent with it, reward desired behaviors, and discourage undesirable ones [18]. Management cannot downplay their informed commitment because most change failures involve a lack of it. Indications of a lack of commitment include the following.

- Organizations do not follow processes in times of crisis.
- Organizations postpone or reduce work to accommodate schedule slips.
- Organizations do not specify quality goals.
- Organizations do not provide training to enhance critical skills.

Sometimes an organization will have to significantly change its structure to correct inadequate management commitment. Executive management may need to realign the entire organization so that software personnel have the absolute responsibility, accountability, and authority over software personnel and development processes. For example, one company's president required process improvement across the entire company and made all software personnel report to the corporate vice president for systems and software engineering, which was a newly created position [46]. Another mature software development company also made the software organization manager the focal point of the software process improvement effort and asked the entire organization to accomplish software process improvement as part of their normal job [20].

Create a software engineering steering group. To foster change, an organization must create an active software engineering steering group composed of executives and senior line management [30]. Line managers contribute resources to a software engineering process group and provide input to it as interested customers. Since line managers ultimately are responsible for implementing change, key line managers should help sell changes. The executives of a software engineering steering group, on the other hand, champion the software process improvement effort, identify improvement targets, assign resources, monitor change processes, and remove progress barriers. A chief executive officer generally leads a software engineering steering group. If not, another senior level *fixer* must head the software process improvement effort and have the authority and resources to negotiate among those affected by a change [30].

The software engineering steering group must adopt an enlightened approach to software process improvement, which is one of three basic approaches to implementing change. The autocratic approach advocates using top leadership to define a change, which an organization enforces using compliance mechanisms. This approach assumes

that there is no need to address human and organizational issues even though mandates are rarely successful and often delay the acceptance of a change [49]. Another approach uses top leadership to determine what needs to change, who then selects a group of people to develop and apply key strategies to align the organization and gain commitment from people. While more effective than the autocratic approach, this approach often falls short of achieving the desired results because it relies on the wisdom and skills of a small number of strategic-thinking and influential people. The enlightened approach, on the other hand, uses stakeholders affected by the change to identify the new behavior and develop transition strategies [27, 36]. This approach requires more people and time to define solutions, but results in a larger group *owning* them [15, 21, 35, 41, 44]. It also capitalizes on the wisdom of senior leadership as well as other people throughout the organization, which results in a greater probability of success than other approaches. Consequently, a software process improvement effort should follow the enlightened change approach.

Create a software engineering process group. Such a group should be chartered to continually improve software engineering processes, and should be given the authority to enforce them [11, 41]. Since its personnel are vitally important to successful change, they should be highly respected people [21, 23, 44, 47] having software process expertise [8, 13, 21, 25]. If such people lack, or are perceived to lack, such expertise and respect, realistic and practical results will not occur because they will lack the necessary credibility to succeed [30, 39]. Consequently, an organization should consider receiving nominations from the practitioners regarding the composition of the software engineering process group.

A software engineering process group should comprise about four percent of the initial workforce and about two percent of the sustaining workforce.[1] The NASA Goddard Space Flight Center actually imposed a ten percent overhead cost on projects to support the NASA Software Engineering Laboratory for collecting and processing data, although these costs initially were only five percent and were later reduced to less than two percent [3]. However, when including data analysis, total costs remained around ten percent [3]. In the opinion of the founders of the NASA Software Engineering Laboratory, it would not have succeeded without this resource commitment [3].

A software engineering process group should consist of both full-time people, to maintain continuity, and practitioners, who understand the processes that an organization actually performs [15, 39]. A good policy is to maintain a staff whose composition is one-third software process experts, who are permanent members of the team, one-third software development experts, who are elected by the practitioners and serve three-year terms, and one-third practitioners, who serve one-year internships whose purpose

1. Frank McGarry, software process improvement presentation, Jet Propulsion Laboratory, 2002.

is to indoctrinate them in practical software engineering and process improvement ideas.

The practitioners should actively participate about half time in the software engineering process group while they work on real projects. The rationale for this approach is that successful technology transfer occurs when personnel have software process knowledge and personally transfer that knowledge to their home organizations. Another way of looking at this is that a software engineering process group is centralized, yet supported by local improvement staffs [12, 41]. This approach is consistent with how the NASA Software Engineering Laboratory deployed new technology to projects [3]. It actively sought *customers*, learned what their needs were, and then tailored the institutional software development process to best fit their needs. In other words, Software Engineering Laboratory personnel became active participants in, or informal members of, project development teams.

When selecting the personnel of the software engineering process group, an organization should choose only those individuals that have demonstrated a commitment to process improvement. This is because many organizations have software process improvement teams that continue to act immaturely [39, 44] even though they are advocating changes to improve maturity. Typical problems of such groups include lack of defined processes, realistic and specified requirements, project plans, defined milestones, and schedules. In other words, software engineering process groups should be run like a software development project. Last, an organization should heed the advice of others who have successfully transformed organizations into higher levels of maturity.

Align human resources. The human resources function of an organization could have the most significant impact on a software process improvement program because people are the most important resource that an organization has. Unfortunately, this fact has largely gone unnoticed within the software process improvement field. Consequently, the following activities are seldom discussed, although they could significantly enhance the ability of an organization to improve its software development processes. In fact, ignoring these issues will almost certainly lead to failure.

- Hire a resilient workforce compatible with the characteristics of the desired organizational culture. Resilient people are positive, focused, flexible, organized, and proactive. Resilient people absorb disruptive change while displaying minimal dysfunctional behavior [10]. These people help develop an adaptive culture, whose characteristics are its focus on innovation, willingness to assume risk, ability to eliminate and correct problems, focus on long-term goals, and willingness and enthusiasm to change. To select these kind of people, the human resources department should develop hiring criteria that evaluate character, self-discipline, motivation, education, training, and skills [34]. The benefits of such selective criteria are improved performance and lowered personnel turnover.

- Generate thorough job descriptions. Job descriptions typically identify the required education, knowledge, and skills of a person that will assume a position, as well as the duties and responsibilities that the person will perform. However, job descriptions seldom specify the rationale for creating the position, or the linkages between or among positions and across functions. Without such information, the person performing the job will be unaware of the organizational agenda and may not operate in a manner consistent with the goals of the entire enterprise [18, 34]. Therefore, the goals and objectives of every job description must be traced to a vision statement [26]. This is because personnel that are not traceable to a vision statement provide no value to the institution and should be eliminated from it or given a new role – one of value to it. Hence, every employee at the institution should have a job description that identifies the rationale for the position, the work relationships of the position, and how the position helps to achieve the institutional goals and objectives. If process improvement is an objective of the organization then every job description must identify process improvement expectations and measurements [16]. Furthermore, to be an effective team player, a person must know the objectives of the team as well as his own. Consequently, a job description should identify the direct suppliers to and customers of the team, the resources that the team consumes and manages, and the objectives of the team, which are aligned with the corporate objectives. Finally, each objective should have quantifiable team and individual performance criteria. In sum, when a job description includes this information, it will be clear what the job contributes to an organization, which will increase employee satisfaction and help it to be cost-effective.
- Establish a defined career path for software professionals [17, 26] based on education, certifications, training, experience, and publication record. Cynicism often surrounds institutional promotions because the criteria are usually qualitative and subjective instead of quantitative and objective, which is contrary to a basic tenet of process improvement – measuring effectiveness – and adversely affects employee morale.
- Provide a problem-solving-based training program since employee competence should be a part of any strategic plan. Thus, organizations should be less interested in improving specific, transient skills and more interested in improving core, enduring skills [18], such as analytical thinking. Analytical thinking helps people to better identify, understand, and solve problems and creates an adaptive workforce that can achieve changing business needs and adopt new technology [4, 16, 26, 34]. Thus, organizations need training curriculums that emphasize analytical thinking, as well as providing training in the latest tools and techniques [17, 21, 26, 29, 35]. If an organization does not provide such training, old practices will render new organizational changes meaningless and a business's operating units will not function more efficiently or effectively.
- Align the reward system with the objectives of a software process improvement program and clearly communicate to the workforce through job descriptions the behaviors that will be rewarded as statements of accountability [17, 34, 39, 44]. These behaviors must provide tangible value to the organization, as defined by a vision statement [18]. In addition, an organization should assess individuals based

on both individual and team performance, which implies that ranking individuals against one another is counter-productive [45]. Finally, an organization should use objective, fair, and open criteria for merit increases and promotions that substantially differentiate pay between top, middle, and low performers [26, 45].

2.2 Responsibilities of the SEPG

Periodically assess the organization. An assessment is a self-audit that provides information to help an organization establish a baseline and define its priorities. An assessment should describe how an organization functions, identify improvement opportunities, and provide recommendations regarding a course of action. Completing a thorough assessment of an organization will help the software engineering process group anticipate how a change will affect the organization and how it should respond to the change. The results of an assessment will shape the general approach for the change by determining initiatives and strategies that will move the organization to the desired state.

Since a software engineering process group should assess the ability of an organization to adopt a change, it should identify the following [31].

- The method an organization has used to handle past change since it affects the success of future changes.
- The activities of past software process improvement efforts that worked well and contributed to success, as well as those that did not.
- The frequency of change that the organization has undergone during the past three years and the dominant effect these changes had on members of the organization.
- Employees' view of the most recent organizational change, in addition to the prevailing attitude toward change.
- Whether employees have understood new policies in the past and if these policies met their needs.
- Employees who will perceive a software process improvement effort as a threat to job security, social status, or power since these people most likely will resist change.

Identify changes and their scope. A software process improvement effort should address the most promising improvement activities for each business unit [27]. For most business units, improvements in their project management, requirements engineering, and validation and verification activities generally yields the greatest benefit. Regardless, an organization should perform four tasks to identify improvement activities.

1. It should focus a software process improvement effort on new or important projects that can most benefit from change [14, 37].
2. It should identify what the customer values most and determine what processes, policies, or strategies it could change or improve to add customer value [37].

3. It should identify strategic and core competency processes that it can improve [7, 20, 46].
4. It should analyze global inefficiencies caused by redundancies performed within activities of the overall business process.

Since an organization needs a gradual approach to change [13, 27, 29, 30, 39], it should change no more than three processes at any given time [9] and introduce changes in three to four month time frames. The reason for this is to prevent personnel from becoming confused about the goals and objectives of the effort or exceeding their emotional limit to change. By limiting the amount of concurrent change, an organization will improve its ability to change because of the clarity presented by a small number of goals and objectives.

In addition, an organization must develop and document a vision for each process it expects to change [16, 32]. These vision statements must describe the new capabilities of the process and identify realistic performance and quality improvement expectations [35]. Further, they must identify how an organization will support the changed processes, respond to customer needs, and respond to competition. A new vision must include measurable objectives for each new process that illustrate dramatic improvement, and fact-based analysis must drive the vision. By creating a vision for each process undergoing change, an organization provides employees with a sense of how they will perform work in the future, which will lower resistance to change by allaying fears that arise when people are uncertain about their futures.

Demonstrate a financial benefit for each change. The importance of demonstrating a beneficial and realistic financial return on investment has been noted by several researchers [6, 21, 27]. The best way to demonstrate a financial benefit is to address the needs of the organization by improving important products; improving processes is relevant only if they have measurable positive impact on product improvement [3]. The NASA Software Engineering Laboratory, for example, was not interested in improving a key process area or achieving higher levels of a process maturity model if such improvement had no demonstrated improvement on product quality. In sum, "[p]roduct measures rather than process changes must be the defined measures of improvement [3]."

Unfortunately, most software process improvement efforts attempt to demonstrate the value of the effort in measured terms of lowered number of defects, increased productivity, and shortened schedules. The problem with this approach is that these measures are difficult for managers, especially high-level executives, to translate into financial value. Worse, no simple measure is provided that permits these decision-makers to compare the value of a software process improvement effort with another competing program. In addition, these measures do not provide a means for determining whether the rate of return is greater than some minimal level that the organization expects to obtain from every investment. This is what most software professionals forget – a software process improvement effort is an investment! Therefore, if an organization has

determined that it wants a return of eight percent on every investment then a software process improvement program must convincingly demonstrate that it will achieve at least that much. If not, it will not be able to acquire or sustain a commitment from executive management, unless it is forced to do so by its customers.

The most sophisticated, and most appropriate, model for computing return on investment is the discounted-cash-flow model because it considers the time value of money. That is, it discounts future cash flows to their present values. To use this model, one first assumes some minimum desired rate of return, set by an organization to cover risk and the cost of money, and then calculates the rate of return, as specified by the model. If the sum of the discounted values of future cash flows is greater than the payout then the project is viable; otherwise, the project costs more than it returns. For every viable project, the excess of the present value over the payout indicates its merit, which an organization can use to compare one proposal with another. This, of course, assumes that there is a limitation of funds and that all beneficial proposals compete with one another.

To perform this computation, the personnel of a software process improvement effort must translate defect and productivity improvements into financial terms. Initially, they must prepare defect and productivity rates and costs based on available data from other organizations. It is best to use data from within the same industry or from those who work on similar problems. Then, the team should revise these rates and costs based on cultural and personnel issues. Be aware that an organization's executives will expect justification for the proposed expected defect and productivity rates and costs.

Obtain the commitment of practitioners. Software organizations that deploy new methodologies often encounter significant resistance [33]. Therefore, software development organizations need to understand the four factors that influence the adoption of a methodology change [42]. A person is likely to adopt a technology if he believes (1) it will enhance his job performance, (2) is voluntary, (3) is consistent with his existing values, needs, and past experiences, and (4) is important to those he respects. Because the usefulness and compatibility of a new methodology with existing work processes has significant influence [34, 41], bottom-up change has been accepted by practitioners as a standard way of doing business [38].

Of all the pressures than an organization can use to impose change, the measurement and reward structure are probably the most important [42]. Therefore, the more strongly practitioners feel that a technology will not help them achieve higher performance levels or become more productive the more important it is for an organization to change the measurement and reward systems to reinforce the desired change [42]. This is especially true during a change in technology because the benefits of such a change are generally not apparent for a long time, whereas the costs incurred by the practitioners are immediate and obvious.

Another effective method for obtaining commitment of the practitioner is by constantly engaging them. Communication is one of the most effective tools an organization can use to obtain acceptance of a change [27, 32, 44]. Hence, it should occur frequently [26, 48] and be performed in one-on-one or in small group situations [30]. Consider, for example, that the CEO of one firm spent six months communicating and justifying his vision to small focus groups when he wanted it to produce personal computers [9].When communicating with the workforce, the organization should perform the following actions to build trust and foster positive change [18].

- Explain the shortcomings of a change followed by its benefits [29] since presenting a balanced view of the change will help to build trust with those affected by a change.
- Listen to the workforce and encourage feedback.
- Inform employees how the software process improvement effort will affect them, honestly, simply, and straightforwardly [27].
- Communicate the result of a change to the affected personnel [45].
- Tailor messages to target audiences [5, 16].

When tailoring a message, an organization must be mindful of the motivators for key groups [2]. Practitioners are motivated to change when they see visible success of software process improvement initiatives, are given adequate resources, and observe successful initiatives that peers initiate. Project managers are motivated to change when they see visible success of software process improvement initiatives, are given adequate resources, and own the software development process. Senior managers, on the other hand, are motivated by the visible benefits of a software process improvement effort and the increased ability of projects to meet targets.

Measure personnel productivity and product quality. An organization should use metrics to measure its progress in transitioning to a desired state [11, 22, 38, 39, 46] because it permits it to compare the rate of actual change against its planned change. These metrics should measure the effectiveness and efficiency of each process, the acceptance of new or changed processes, and observed product quality [16, 18]. These measurements should form a hierarchy, such that the higher-level measurements relate directly to organizational goals and the lowest level goals relate to detailed individual measurements, such as total labor hours per developed line of code. Thus, the movement of lower-level indicators will predict movement of the higher-level indicators.

3 Summary

Proponents of software process improvement claim improved product quality, reduced time to market, better productivity, increased organizational flexibility, and greater customer satisfaction. Some have even reported that organizations can reduce defects by 95 percent and schedules by 71 percent and increase productivity by 222 percent [1]. Regardless of these potentialities, about two-thirds of software process improve-

ment efforts fail [1]. Some organizations have even found it difficult to establish simple metrics programs [24].

This paper has proposed several non-technical practices to improve the success of software process improvement efforts. The basic premise is that to affect change, software organizations must address both social and technological aspects. For an organization to succeed with software process improvement, it must successfully achieve four critical objectives. Failing to successfully perform any of these objectives will cause the software process improvement effort to fail. The first critical objective is constructing the vision of the new organization and the individual changed processes. Unfortunately, this is seldom addressed by software process improvement plans nor discussed by software process improvement professionals. This is somewhat strange considering that modern software development processes usually specify the development of a concept of operations, which essentially outlines the vision of a proposed software system. So why is it that software process improvement professionals think that they can create a new system – a software process – without doing what they themselves advocate to software practitioners? Thus, a software process improvement program must develop an operational concept and support it with several operational scenarios. Besides explaining what the future will look like, operational scenarios may also help to identify who will resist change [30].

The second critical objective is to obtain commitment at all organizational levels [1]. Obviously, the most desired form of commitment is to have employees identify and be involved with a software process improvement effort because other forms of commitment are only sustained through reward or organizational control systems. However, these methods are useful for sustaining commitment when commitment is adversely affected by current attitudes, circumstances, and organizational factors [1].

Obtaining executive-level commitment for a software process improvement effort is the most important of all levels to achieve. Without high-level commitment, the commitment of others most likely will not be forthcoming. To obtain executive-level commitment, a software process improvement team must successfully demonstrate how the software process improvement effort supports the organizational strategy and business needs. For example, if the addressed business need is to reduce software development costs by ten percent then the software process improvement effort must demonstrate how it will reduce those costs.

The third critical objective is to involve practitioners in the development of the software process improvement initiative. Such involvement should include ten to twenty-five percent of the workforce, depending on the size of the organization. Without such involvement an organization will not be able to gain acceptance of a software process improvement effort because practitioners will resist what they did not help to create. Even if mandated, they will not necessarily follow the process. If they do follow a new process, they will do so to simply satisfy a mandate, which will not provide the desired benefits.

The fourth critical objective is to clearly communicate the software process improvement effort – the vision, its benefits, its differences, and so on – to the entire workforce. The software engineering process group will have to do this for executives, middle-level managers, and practitioners. For each of these groups, the software engineering process group should develop a different message. For high-level executives, the message should concentrate on the alignment of the software process improvement effort with the business goals and needs of the organization and the financial return on investment. For practitioners, the message should emphasize the difference between how they do work today and how they will do it in the future. The software engineering process group should emphasize the personal benefits of the change, and emphasize tedious operations that practitioners currently have to perform manually that will be automated in the future. Finally, the software engineering process group should motivate middle-level managers by emphasizing the benefits in terms of improved product quality, reduced schedules, and greater predictability of schedules. That is, the software engineering process group should emphasize that managing projects will be easier in the new state than in the prior one.

In conclusion, this paper has identified numerous practices for changing an organization. Organizations should adopt these practices to improve their software process improvement efforts. To promote success, an organization must address four critical objectives – creating a vision of the future organization, achieving executive commitment, involving practitioners in change definition, and communicating that vision to the entire organization. In addition, a select group of people, having the necessary skills and personal characteristics, should lead and participate in software process improvement efforts.

Acknowledgments

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

1 Abrahamsson, P.: Commitment Development in Software Process Improvement: Critical Misconceptions. Proceedings of the International Conference on Software Engineering (2001) 71-80.

2 Baddoo, N. and Hall, T.: Motivators of Software Process Improvement: an analysis of practitioners' views. Journal of Systems and Software **62** (2002) 85-96.

3 Basili, V. R., McGarry, F. E., Pajerski, R., and Zelkowitz, M. V.: Lessons learned from 25 years of process improvement: The Rise and Fall of the NASA Software Engineering Laboratory. Proceedings of the International Conference on Software Engineering (2002) 69-79.

4 Beer, M., Eigenstat, R.A., and Spector, B.: Why change programs don't produce change. Harvard Business Review **68**:November-December (1990) 158-166.

5 Berger, L. A.: Change Management. In: Berger, L. A. and Sikora, M. J. (eds.): The Handbook of Change Management: A Road Map to Corporate Transformation. Irwin. (1994) 3-23.
6 Brodman, J. G. and Johnson, D. L.: Return on Investment (ROI) from Software Process Improvement as Measured by US Industry. Software Process: Improvement and Practice **1**:1 August (1995) 35-47.
7 Burrill, C. W.: Ten Ways to Kill a Quality Program. IEEE Engineering Management Review **24**:1 Spring (1996) 105-108.
8 Butler, K. and Lipke, W.: Software Process Achievement at Tinker Air Force Base, Oklahoma. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-2000-TR-14 (2000).
9 Carr, D. K. and Johansson, H. J.: Best Practices in Reengineering: What Works and What Doesn't in the Reengineering Process. McGraw-Hill (1995).
10 Conner, D. R.: The Next Generation of Fire Walkers. In: Berger, L. A. and Sikora, M. J. (eds.): The Handbook of Change Management: A Road Map to Corporate Transformation. Irwin (1994) 257-269.
11 Crosby, P. B.: Quality is Free. McGraw-Hill (1979).
12 Curtis, B.: The global pursuit of process maturity. IEEE Software July/August (2000) 76-78.
13 Debou, C. and Kuntzmann-Combelles, A.: Linking software process improvement to business strategies: experiences from industry. Software Process: Improvement and Practice **5**:1 March (2000) 55-64.
14 Diaz, M. and Slago, J.: How software process improvement helped Motorola. IEEE Software **14**:5 (1997) 75-81.
15 Dion, R.: Process improvement and the corporate balance sheet. IEEE Software **10**:4 (1993) 28-35.
16 Eckes, G.: Making Six Sigma Last. Wiley (2001).
17 Ferguson, P., Leman, G., Perini, P., Renner, S., and Seshagiri, G.: Software process Improvement Works!. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-99-TR-27 (1999).
18 Fitz-enz, J.: The 8 Practices of Exceptional Companies: How Great Organizations Make the Most of Their Human Assets. AMACOM (1997).
19 Goldenson, D. R. and Herbsleb, J. D.: After the Assessment: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-95-TR-009 (1995).
20 Haley, T., Ireland, B., Wojtaszek, E., Nash, D., Dion, R.: Raytheon Electronic Systems Experience in Software Process Improvement. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-95-TR-17 (1995).
21 Hall, T., Rainer, A. and Baddoo, N.: Implementing software process improvement: an empirical study. Software Process: Improvement and Practice **7**:1 March (2002) 3-15.
22 Hammer, M.: Beyond Reengineering. HarperCollins (1996).
23 Herbsleb, J. D. and Goldenson, D. R.: A systematic survey of CMM experience and results. Proceedings of the International Conference on Software Engineering (1996) 323-330.
24 Herbsleb, J. D. and Grinter, R. E.: Conceptual simplicity meets organizational complexity: case study of a corporate metrics program. Proceedings of the International Conference on Software Engineering (1998) 271-280.
25 Humphrey, W. S., Snyder, T. R., and Willis, R. R.: Software process improvement at Hughes Aircraft. IEEE Software **8**:4 (1991) 11-23.

26 Humphrey, W. S.: Managing technical people: innovation, teamwork, and the software process. Addison-Wesley (1997).
27 Juran, J. M. and Gryna, F. M.: Quality Planning and Analysis: From Product Development through Use. McGraw-Hill (1993).
28 Kabat, D. J.: Information Technologies to Manage the Next Dynamic. In: L. A. Berger and M. J. Sikora (eds.): The Change Management Handbook: A Road Map to Corporate Transformation. Irwin (1994).
29 Kaltio, T. and Kinnula, A.: Deploying the defined SW process. Software Process: Improvement and Practice **5**:1 March (2000) 65-83.
30 Keen, P. G. W.: Information Systems and Organizational Change. Communications of the ACM **24**:1 January (1981) 24-33.
31 Kitson, D. H. and Masters, S. M.: An analysis of SEI software assessment results. Proceedings of the International Conference on Software Engineering (1993).
32 Kotter, J. P.: Leading change: why transformation efforts fail. Harvard Business Review **73**:2 (1995) 59-67.
33 Kozar, K.: Adopting Systems Development Methods: An Exploratory Study. Journal of Management Information Systems **5**:4 (1989) 73-86.
34 Labovitz, G. and Rosansky, V.: The Power of Alignment. Wiley (1997).
35 Laporte, C. Y. and Trudel, S.: Addressing the people issues of process improvement activities at Oerlikon Aerospace. Software Process: Improvement and Practice **4**:4 December (1998) 187-198.
36 Lawrence, P. R.: How to deal with resistance to change. In: Management of Change, Harvard Business Review (1991) 77-85.
37 March, A.: A Note on Quality: The Views of Deming, Juran, and Crosby. IEEE Engineering Management Review Spring (1996) 6-14.
38 McGarry, F., Page, G., Waligora, S., Basili, V., and Zelkowitz, M.: Software Process Improvement in the NASA Software Engineering Laboratory. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-94-TR-22 (1994).
39 Moitra, D.: Managing change for software process improvement initiatives: a practical experience-based approach. Software Process: Improvement and Practice **4**:4 December (1998) 199-207.
40 Pitterman, B.: Telcordia technologies: the journey to high maturity. IEEE Software **17**:4 (2000) 89-96.
41 Rainer, A. and Hall, T.: An analysis of some core studies of software process improvement. Software Process: Improvement and Practice **6**:4 December (2001) 169-187.
42 Riemenschneider, C. K., Hardgrave, B. C., and Davis, F. D.: Explaining Software Developer Acceptance of Methodologies: A Comparison of Five Theoretical Models. IEEE Transactions on Software Engineering **28**:12 December (2002) 1135-1145.
43 Rifkin, S.: Why software process innovations are not adopted. IEEE Software July/August (2001) 110-112.
44 Stelzer, D. and Mellis, W.: Success factors of organizational change in software process improvement. Software Process: Improvement and Practice **4**:4 December (1998) 227-250.
45 Weinberg, G. M.: Understanding the Professional Programmer. Dorset House (1988).
46 Willis, R. R., Rova, R. M., Scott, M. D., Johnson, M. I., Ryskowski, J. F., Moon, J. A., Shumate, K. C., and Winfield, T. O.: Hughes Aircraft's Widespread Deployment of a Continuously Improving Software Process. Software Engineering Institute, Carnegie Mellon University, Technical Report CMU/SEI-98-TR-6 (1998).

47 Wilson, D. N., Hall, T., and Baddoo, N.: A framework for evaluation and prediction of software process improvement success. Journal of Systems and Software **59**:2 (2001) 135-142.
48 Wohlwend, H. and Rosenbaum, S.: Schlumberger's software improvement program. IEEE Transactions on Software Engineering **20**:11 (1994) 833-839.
49 Zelkowitz, M. V.: Software engineering technology infusion within NASA. IEEE Transactions on Engineering Management **43**:3 (1996) 250-261.