The Past, Present, and Future of Software Process Improvement

Carnegie Mellon University Stephen E. Cross

1. Introduction

During the past several years, organizations have experienced a revolution in how they work driven largely by the demands of their customers and the desire to be the first in a market with a new product. Software professionals have been focusing attention on process improvements for more than 20 years. Organizations that have committed to and practiced software process improvement (SPI) have reaped its rewards.[2] They recognize that success in delivering a quality product on time and on schedule is inextricably linked to the experience of their organizations. By using better software engineering methods, they have found ways to capture, share, and improve their knowledge and experience.

2. Software Process Basics

A *process* is a sequence of steps, actions, or activities that members of an organization perform to bring about a desired result or achieve a goal. We at the Software Engineering Institute (SEI) also consider processes to be leverage points for an organization's sustained improvement.[3]

The SEI's Capability Maturity Models[4] use the concept of *process* areas, which are the basic building blocks of maturity models.[5] A process area does not describe how an effective process is executed (e.g., entrance and exit criteria, roles of participants, resources). Instead, a process area describes what the people who are using an effective process do (their practices) and why they do those things (their goals). Process areas describe key aspects of such processes as configuration management, requirements management, product verification, systems integration, and many others.

3. Survey of Process Modeling Approaches

The use of Capability Maturity Models is one approach to software process improvement, but there are numerous other models and assessment methods. SPICE, Bootstrap, TickIT, and the standards emanating from the International Organization for Standardization (ISO) are major examples.

SPICE(Software Process Improvement and Capability Determination)

The SPICE Project originated in 1992 when the standardization community began developing an international standard for software process assessment. The project resulted in the ISO 15504 standard, published in 1998. The methods and models used to develop SPICE were, most notably, the Capability Maturity Model; Bootstrap, developed by the Bootstrap Institute; Trillium, developed by Bell Canada, Northern Telecom, and Bell Northern Research; and SPQA, developed for internal use within Hewlett Packard. The SPICE standard is now taught in software engineering curricula. Hundreds of people have been trained in

SPICE training courses, and a certified SPICE assessor program exists. There is also a Korean version of SPICE, called "KSPICE."[6]

Bootstrap

Bootstrap is a European software process assessment and improvement method based on the Capability Maturity Model, the ISO 9000 series of standards, and the Software Engineering Standards of the European Space Agency.[7]

While supporting the basic concepts of the CMM, Bootstrap provides some more detailed capability profiles that organizations can use to identify important areas for further improvement that are not easily uncovered by other methods. The method has been effective in helping organizations to assess their current status of software process quality and to initiate appropriate improvement actions. An ISO 15504-compliant version of Bootstrap has been developed by the Bootstrap Institute.

TickIT[8]

The British Department of Industry and Trade sponsored development of the TickIT scheme to help address problems with software quality. TickIT involves a system of accredited certification bodies, a means of ensuring that auditors are appropriately qualified, and the publication of the TickIT Guide, which incorporates ISO 9000-3, the ISO guide applicable to service organizations.

4. Industry Adoption

Among the numerous successful SPI methods that have emerged over the past decade, the SEI naturally has the most experience and insight into the Capability Maturity Model, and how it has been used in practice.

Since 1990, more than 5,000 organizations[9] have invested in CMM-based software process improvement. These efforts have included

- in-depth assessment of current practices;
- education and training in industry best practices and transition techniques; and
- implementation of improvements consistent with the organizations' business objectives.

A broad spectrum of organization types is represented among the 5,000 organizations, including manufacturing, services, utilities, trade, and administration. About 32% are non-U.S. organizations. Also, CMM-based SPI is not limited to large organizations. More than 47% employ 100 or fewer people. About 23% employ between 100 and 200. Some 29% employ more than 200.[10]

SEI data show that after organizations implement CMM-based improvement, median annual productivity improves by 35%, time-to-market is reduced by 19%, and post-release defects drop by 39%. The median annual cost per engineer of software process improvement using the CMM for Software (SW-CMM) is $1,375. The savings to organizations are about five times this amount.

Our experience with three corporations, Boeing, Lockheed Martin, and Motorola, provides examples of what has been accomplished with the SW-CMM. At Boeing, the productivity of projects increased by 62% as those parts of the organization responsible for projects achieved Maturity Level 3. Cycle times improved by 36%, planning was more accurate, defects could be detected much earlier in the process, and product quality increased. Both customer and employee satisfaction also increased as the organization moved up the capability ladder.

More specifically, at Boeing Space and Transportation Systems, which has been assessed at SW-CMM Level 5, defects are nearly all found and fixed before testing begins. Defects escaping into the field have been reduced from 11% to practically 0%. Programs consistently reach customer satisfaction and performance targets. And, while peer reviews increase total project costs by 4%, rework during testing is reduced by 31%. The return on investment is 7.75 to 1.[11]

At Lockheed Martin's Manassas unit, which achieved SW-CMM Level 5 in February 1999, errors have declined and productivity has increased by 80 percent. Performance measures have improved and converged with higher maturity. Lockheed Martin's Owego unit saw productivity gains of 452.9% between 1982 and 2000, as productivity improved with each maturity level attained.[12]

Motorola "has long been a champion of the SEI's CMM as a vehicle for furthering software process improvement," in the words of two top Motorola software executives.[13] As Motorola improved its capability, the company's cost, cycle time, and defect density dropped sharply, while quality and productivity improved dramatically. "Achieving [the CMM Level 5] rating provides our customers with the assurance that they are receiving high-performance solutions that improve operations across the enterprise," according to Leif Soderberg, Motorola senior vice president.

5. Globalization

Today, nearly every organization everywhere in the world that acquires or manufactures a product or delivers a service must be concerned with software quality. Indian software companies provide an excellent example of how adoption of software process improvement methods can help organizations rapidly establish world-class software capabilities.

Simply stated, Indian software companies and software professionals have a quality mindset. According to Satish Bangalore, managing director of Phoenix Global Solutions India, "It's almost shameful for them to admit they are a Level 2 company or that they didn't get ISO 9000 certification on the first or second attempt."[14] A major Indian software development firm, Infosys Technologies Ltd., has embraced CMM, ISO, Six Sigma, and the Malcolm Baldrige National Quality Award framework. The company measures and manages such things as in-process defects, rework costs, defects delivered to customers, cost overruns, schedule slippage, and estimation accuracy.

There are now an estimated 150,000 information-technology professionals in India-60,000 software engineers, 15,000 graduates in computer science and related disciplines, and 75,000 from other engineering disciplines. For the past decade, the Indian software industry has been influenced by standard approaches to project management, such as ISO 9000 and the SW-CMM. Indian software firms embraced quantitative process management and many have now achieved Level 3 or 4 in formal CMM-based assessments. Software exports have doubled over the past five years. Many North American firms have expanded operations, and others, such as American Express and Oracle, have opened new operations.

Top Indian software firms are no longer satisfied with just writing code, but are increasing their capabilities in architecture and whole systems design, systems integration, and systems migration.[15]

Software development costs in India are currently about one-third those in the United States, but that advantage is expected to disappear in three to five years. Then, Indian firms will compete on quality alone-and they appear to be fully capable of doing so. Estimates call for annual revenue for India's software industry to grow from $5.7 billion in 2001 to $87 billion in 2008.

6. Evolving Standards

Some new standards are evolving as a result of the work of the international standards community. In 1995, ISO/IEC 12207, Information Technology-Software Life Cycle Processes was published. This standard describes the processes for acquiring, developing, supplying, operating, and maintaining software. In 1998, the U.S. implementation of this standard was published as IEEE/EIA 12207, and includes additional material based on U.S. best practices.[16] IEEE/EIA 12207 provides an architecture of the full software life cycle, including product conception, implementation, maintenance, and software retirement, and specifies the processes, activities, and tasks to be accomplished during the life cycle.

CMM-based software process improvements can give organizations a leg up in meeting the requirements of 12207. Most organizations operating at CMM Level 3 or higher are very close to being compliant with 12207 simply by achieving Level 3. Organizations that have not yet used the CMM approach to SPI can use it as a method for achieving 12207 compliance because the CMM provides an incremental, five-level approach to improvement, and step-by-step guidance.

ISO/IEC 15504, an emerging international standard on software process assessment, resulted from the SPICE Project of the 1990s.[17] It defines a number of software engineering processes, such as software requirements analysis, and a scale for measuring an organization's capabilities. As with the CMM, a basic premise of the measurement scale is that higher process capability is associated with better project performance.

7. Related Trends

Two significant trends have emerged during the past few years. First there has been a proliferation of process models for software engineering, systems engineering, and other disciplines. The SEI, in collaboration with industry and the U.S. government, has recently published an integrated framework called CMM Integration (CMMI) in order to harmonize models, training, and assessment approaches. Second, there has been a strong desire to accelerate the time it takes to achieve higher maturity levels under a CMM and to apply CMM concepts to the individual engineer and small teams. The SEI has developed the Personal Software Process (PSP) and the Team Software Process (TSP) to meet this need. In addition, while not discussed in this paper, we have recently seen the rise of the "light process" movement as typified by Extreme Programming[18] (XP). Many in the "Internet development" and rapid-application-development communities view the use of models such as the SW-CMM as a "heavy" and burdensome approach to process improvement. As discussed by Paulk[19] and Leishman,[20] XP and similar approaches are best practices applied to code development by small teams and are therefore worth consideration and compatible with the SW-CMM.

CMM Integration[21]

Capability Maturity Models have proliferated along with the growth of software-intensive systems. These trends have resulted in a blurring of the line between software and systems development. It can be helpful to retrace the growth of CMMs, to understand why an integration of these models is now desirable.

In 1991, the Software Engineering Institute released the Capability Maturity Model for Software (SW-CMM) consisting of key practices organized into a "roadmap" that guides organizations toward improving their software development and maintenance capability. The SW-CMM approach was based on principles of managing product quality that have existed for the past 60 years. In the 1930s, Walter Shewhart advanced the principles of statistical quality control, which were further developed and successfully demonstrated in the work of W. Edwards Deming, Joseph Juran, and Philip Crosby.

The SEI eventually became involved in helping to develop additional CMM approaches in other disciplines. The models that emerged from these efforts include the Systems Engineering Capability Maturity Model (SE-CMM) and the Integrated Product Development Capability Maturity Model (IPD-CMM). As is the case with the SW-CMM and SE-CMM, the IPD-CMM addresses organizational and project management processes, but with a focus on ensuring the timely collaboration of all appropriate disciplines in the development and maintenance of a product or service.

The development of multiple Capability Maturity Models for other disciplines was generally greeted positively. Process improvement expanded to affect more disciplines and helped organizations to better develop and maintain their products and services. However, this expansion also created challenges.

Ideally, various Capability Maturity Models should work together harmoniously for the benefit of organizations wishing to apply more than one model to improve product quality and productivity. However, especially with respect to the Capability Maturity Models for software engineering, systems engineering, and IPD, managers have found that overlaps in content and differences in architecture and guidance across these models made improvement across the organization difficult and costly. Training, assessments, and improvement activities often had to be repeated for each specific discipline, with little guidance on how to integrate such activities across these disciplines. Organizations needed a way to easily integrate their CMM-based improvement activities. The models themselves needed to be integrated.

To respond to the challenges and opportunities created by the demand for better integration of CMM models, training, and assessment methods, the Office of the U.S. Under Secretary of Defense for Acquisition and Technology initiated the CMM Integration project, which is co-sponsored by the National Defense Industrial Association. Experts from a variety of backgrounds and organizations were asked to establish a framework that could accommodate current and future models.

Since February 1998, industry, government, and the SEI have developed the following CMMI models:

- CMMI for Systems Engineering and Software Engineering(CMMI-SE/SW)
- CMMI for Systems Engineering, Software Engineering, and Integrated Product and Process Development(CMMI-SE/SW/IPPD)
- CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing(CMMI-SE/SW/IPPD/SS)
- CMMI for Software Engineering(CMMI-SW)

CMMI products comply with the ISO 15504 standard for software process assessment, and preserve the improvement work achieved by organizations that used CMMI source models.

The CMMI product suite includes CMMI models, a framework, training materials, and assessment methods. The CMMI framework states the rules and concepts that ensure CMMI products are consistent with each other-that is, that they are capable of being integrated.

Worldwide adoption of CMMI is progressing rapidly. As of this writing, nearly 7,500 people have taken the Introduction to CMMI course-a number that grows by 400 people each month-and more than 450 have taken the Intermediate CMMI course. As of October 2002, 40 CMMI appraisals had been conducted; about 55% of those were at non-U.S. organizations. A substantial number of people have been trained by the SEI to offer CMMI training and appraisals: as of February 2003, the SEI has licensed 50 people to teach CMMI courses and 86 people to offer CMMI appraisal services.

In one example of a large organization's adoption of CMMI, Lockheed Martin issued a new corporate policy stating that it will apply the highest standards of engineering excellence to all projects. As part of the policy, it is requiring each of its business units to attain, by January 2005, at least a CMMI Maturity Level 3 against the CMMI-SE/SW/IPPD/SS model. After an initial appraisal, business units are strongly encouraged to move up to the next-higher CMMI level about every two years, until they reach Maturity Level 4 or 5.

Organizations that are interested in complying with the latest standard for software engineering and product quality from ISO and the International Electrotechnical Commission(ISO/IEC 9126) will find CMMI helpful because it is both a process and product quality model. The product quality attributes of functionality, reliability, security, usability, efficiency, maintainability, and portability are addressed in the ISO/IEC standard and supported by CMMI. Additional SEI work, such as architecture tradeoff analysis, provides technical approaches for achieving assured levels of product quality during design, as opposed to the state-of-practice approach of determining product quality during testing.

Personal Software Process(PSP) and Team Software Process(TSP)

Although the Capability Maturity Model provides a powerful improvement framework, its focus is necessarily on "what" organizations should do and not "how" they should do it. This is a direct result of the CMM's original motivation to support the Department of Defense acquisition community. Developers of the CMMs knew management should set goals for their software work but they also knew that there were many ways to accomplish those goals. Above all, they knew that no one was smart enough to define how to manage all software organizations. Thus, the CMM focused on goals and provided only generalized examples of the practices the goals implied.

As organizations used the CMM, many had trouble applying the CMM principles. In small groups, for example, it is not generally possible to have dedicated process specialists, so every engineer must participate at least part time in process improvement. There was a need for much greater process detail, and greater understanding and emphasis on the real practices of development engineers.

Improvement requires change, and changing the behavior of software engineers is a nontrivial problem: engineers only believe new methods work after they use them and see the results, but they will not use the methods until they believe they work. To convince software engineers of the value of better methods, the Personal Software Process (PSP) requires that they leave their day-to-day environment and go through a rigorous training course.

At the U.S. company Advanced Information Services(AIS)[22] team members were trained in PSP in the middle of a project. Before PSP training they had difficulty producing estimates; in one case, the original estimate was four weeks, but the job took 20 weeks. Their average estimating error was 394 percent. After PSP training, these same engineers' estimating error was -10.6 percent. The same type of results occurred in the area of test defects. Before PSP training, the engineers had a substantial number of acceptance test defects and their products were uniformly late. After PSP training, the next product was nearly on schedule, and it had only one acceptance test defect.

We have found, however, that PSP does not go far enough. Even when everyone on a team of engineers is PSP trained and properly supported, they still must figure out how to combine their personal processes into an overall team process. We have found this to be a problem even at higher CMM levels. This is why the SEI has developed the Team Software Process(TSP).[23]

TSP extends and refines the CMM and PSP methods to guide engineers in their work on development and maintenance teams. It shows them how to build self-directed teams and how to perform as effective team members. It also shows management how to guide and support these teams and how to maintain an environment that fosters excellent team performance.

The principal benefit of TSP is that it shows engineers how to produce quality products for planned costs and on aggressive schedules. It does this by showing engineers how to manage their work and by making them owners of their plans and processes.

While TSP is still in development, the early results are encouraging. One team at Embry-Riddle

Aeronautical University removed more than 99 percent of development defects before system test entry. A team at Hill Air Force Base reduced testing time by eight times and more than doubled its productivity. The team's customer has since found no defects in using the product. ABB Ltd. used PSP/TSP methods and found only .44 defects per thousand lines of code, a 10-times reduction compared to a previous project completed without using the TSP process.

8. Future Trends

Industry leaders are aggressively improving their software engineering practices to drastically reduce the amount of defects introduced during design and coding. In the future, "software product quality" will be less about defects and more about how well a product achieves desired levels of functionality, reliability, security, usability, efficiency, maintainability, and portability. The latest developments in software process improvement, such as CMMI and TSP, the SEI's work in architecture tradeoff analysis, and the ISO/IEC 9126 standard, will accelerate the trend toward this expanded definition of quality.

There are three other future trends that I consider significant, one that is now quite clear, and two upon which I can only speculate. The first has to do with software reuse. The second and third have to do with technology to support the use of software engineering processes and to support virtual organizations.

Software Reuse

Organizations that have instituted processes and as a result know they produce high-quality code have an extreme competitive advantage when they reuse that code. The SEI has developed the Framework for Product Line Practice,[24] which is based on the best practices of organizations that systematically reuse software and associated knowledge and experience (e.g., architecture, requirements, and plans) across a family of similar products. Significant productivity, cycle time, and quality improvements have been noted. More details are in the recent book Software Product Lines: Practices and Patterns.[25] In addition, the IEEE is introducing the IEEE 1517 Software Reuse Standard[26] to provide a guide for best practices in software reuse.

Software Process Technology

During the past five years, several companies have provided computer-based libraries of predefined processes. But there is also an exciting and fruitful area of research aimed at helping organizations discover, represent, and use effective and efficient processes. For example, in Osterweil's research,[27] a process programming language and interpreter are being developed to coordinate the efforts of people, computers, and software tools to support key software development activities, such as collaborative design and software testing and analysis. As Osterweil points out, "This work also provides a platform for research on how to perform rigorous evaluation, comparison, analysis, evolution and improvement of software development processes themselves."

Virtual Organizations

The last trend is currently a necessity in the global economy. Software organizations are now "24x7" (i.e., they work 24 hours a day, 7 days a week). It is not uncommon for a development organization in the United States to be working in partnership with development organizations in Europe and Korea where work is done continuously and collaboratively during the daylight hours in each part of the world. This presents two interesting questions for organizations that have in the past reaped the benefits of process improvement. First, how can process improvement under a model such as CMM be effectively supported in different cultures? The second question is equally intriguing. How can several organizations, each committed to process improvement, come together to work on a project for a period of time and rapidly integrate their processes and cultures in order to perform effectively on the project? While it is premature to attempt an answer to these questions, relevant research is being conducted at the Institute for Software Research International[28] and the Software Industry Center[29] at Carnegie Mellon University.

9. Summary

Software process improvement has been one of the most important ways to improve the quality of software and the effectiveness and efficiency of software development organizations. During the past 10 years, more than 5,000 organizations have adopted the SW-CMM and many other organizations have adopted similar process models. IEEE and international standards are evolving to encourage and support continued process improvement. The Software Engineering Institute is committed to broadening the benefits of software process improvement to the entire engineering organization through a new model called CMM Integration and to enable faster improvement through the Personal Software Process and the Team Software Process. The SEI is also committed to working with its colleagues in Korea so that industry can reap the benefits from this important and exciting work.

References

[1] Much of this paper is based on my keynote presentation to the 2001 Korea Software Engineering Process Group Conference: Cross, Stephen E., "The Value of Software Process Improvement," keynote speech, 2001 Korea Software Engineering Process Group (SEPG) Conference, Seoul, Korea, 20-21 September 2001. Proceedings published by System Integration Technology Research Institute (SITRI), Seoul, Korea, 2001.

[2] McConnell, S., "The Power of Process," Computer, May 1998, pp. 100-102.

[3] Software Engineering Institute. CMMISM-SE/SW, V1.0: Capability Maturity Model(r)-Integrated for Systems Engineering/Software Engineering, Version 1.1, Continuous Representation (CMU/SEI2002-TR001). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute. 2000.

[4] Capability Maturity Model, CMM, and CMMI are registered in the United States Patent and Trademark Office by Carnegie Mellon University. CMM Integration, Personal Software Process, PSP, Team Software Process, and TSP are service marks of Carnegie Mellon University.

[5] Shrum, S., "Spotlight: CMMI Model Representations," SEI Interactive, December 1999. See http://interactive.sei.cmu.edu/Features/1999/December/Spotlight/Spotlight.dec99.pdf.

[6] For more about KSPICE see http://www.kspice.co.kr/html/main.asp.

[7] Stienen, H., Engelmann, F., Lebsanft, E., "BOOTSTRAP: Five Years of Assessment Experience; Software Technology and Engineering Practice," Proceedings of the Eighth IEEE International Workshop on Incorporating Computer-Aided Software Engineering, 1997, pp. 371-379.

[8] Morrison, H., "Standards and Certification," Layman's Guide to Software Quality, IEEE Colloquium, 1993.

[9] Sheard, S, "The Frameworks Quagmire," CrossTalk, September 1997. See http://www.stsc.hill.af.mil.

[10] Software Engineering Institute, Benefits of CMM-Based Software Process Improvement: Initial Results(CMU/SEI-94-TR-013). See http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.013.html.

[11] Yamamura and Wigle, Boeing Space and

Transportation Systems, CrossTalk, August 1997.

[12] Software Engineering Process Group Conference, 1999.

[13] Diaz, M., and Sligo, J., "How Software Process Improvement Helped Motorola," IEEE Software, October 1997.

[14] Anthes, G., and Vijayan, J., "Lessons from India Inc.," Computerworld, April 2, 2001.

[15] Embar, C. "The State of Software Development in India," CrossTalk, August 2001. See http://www.stsc.hill.af.mil.

[16] Ferguson, J., and Sheard, S., (Software Productivity Consortium, "Leveraging Your CMM Efforts for IEEE/EIA 12207," IEEE Software, September/October 1998.

[17] El Emam, K., and Kirk, A., "Validating the ISO/IEC 15504 Measure of Software Requirements Analysis Process Capability," IEEE Transactions on Software Engineering, Volume 26, Issue 6, June 2000.

[18] Beck, K., "Embracing Change with Extreme Programming," IEEE Computer, October 1999.

[19] Paulk, M., "Extreme Programming from a CMM Perspective," Proceedings of the XP Universe Conference, July 2001. See http://www.xpuniverse.com.

[20] Leishman, T., "Extreme Methodologies for an Extreme World," CrossTalk, June 2001. See http://www.stsc.hill.af.mil.

[21] Ahern, D., Clouse, A., and Turner, R., CMMISM Distilled: A Practical Introduction to Integrated Process Improvement, Addison-Wesley, 2001.

[22] Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S., and Matvya, A., "Introducing the Personal Software Process: Three Industry Case Studies," IEEE Computer, Vol. 30, No. 5, May 1997, pp. 24-31.

[23] Humphrey, W. Introduction to the Team Software Process, Addison-Wesley, 2001.

[24] See http://www.sei.cmu.edu/plp/framework.html

[25] Clements, P. and Northrop, L., Software Product Lines: Practices and Patterns, Addison-Wesley, 2002.

[26] McCLure, C. Software Reuse: A Standards-Based Guide, IEEE Computer Society, 2001.

[27] See http://laser.cs.umass.edu/process.html.

[28] See http://spoke.compose.cs.cmu.edu/isri/.

[29] See http://www.heinz.cmu.edu/swic/.

Stephen E. Cross

Since 1996, Stephen E. Cross has been the Director and Chief Executive Officer of the Software Engineering Institute (SEI) at Carnegie Mellon University.
He joined the university in 1994 as a member of the research faculty and Director of the Information Technology Center.
Currently, he holds a joint appointment as a Principal Research Scientist in the School of Computer Science.
He received his Ph.D. from the University of Illinois at Urbana-Champaign, his Master of Science in Electrical Engineering from the Air Force Institute of Technology, and his Bachelor of Science in Electrical Engineering from the University of Cincinnati.
He is a graduate of the U.S. Air Force (USAF) Test Pilot School (Flight Test Engineer Course), the USAF Air War College, and the National Defense University.
E-mail : sc@sei.cmu.edu