Future Directions in Process Improvement

Watts S. Humphrey, Dr. Michael D. Konrad, James W. Over, and William C. Peterson
Software Engineering Institute

As systems become larger and more complex, and as increasing numbers of development programs are integrated and distributed, development processes, methods, and management must change. To keep pace with these changes, the directions of process improvement must also change. This article describes the likely future directions of process improvement evolution.

Large and complex computer-based systems are critical to the economic and military welfare of the United States and to much of the industrialized world. These systems form the backbone of modern military, business, and governmental operations, and without their continued support, our societies would be severely inconvenienced and even threatened. Unfortunately, the development of such systems has been troubled, and the systems needed for the future will be vastly more complex and challenging. If history is any guide, attempting to develop these future systems with current widely followed practices will almost certainly yield unsatisfactory results.

In addressing the challenges of the future, it is important to consider three points. First, with few exceptions, the reasons that large-scale development programs have failed have not been technical [1, 2]. As the cancellation of two large and critical efforts demonstrates, these systems have almost always failed because of program-management problems [3, 4]. Second, the solutions to these program-management problems are known and have proven to be highly effective, but they are not widely practiced. Third, if these known and understood project management practices are not promptly and effectively adopted, future large-scale systems development programs will be completely unmanageable. Such systems will no longer be delivered late, over cost, and with poor quality; they will likely not be delivered at all. Proper use of sound processes would actually improve the cost, quality, and schedule performance of engineering organizations.

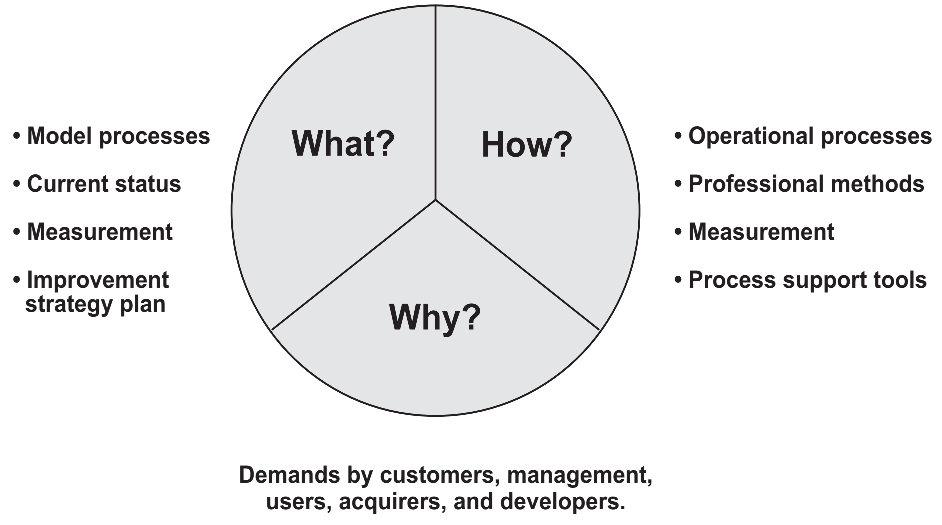
This article outlines the authors' current thinking on the likely future direction of process-improvement work and our strategy for addressing the challenges ahead. To explain the objectives of this article, however, it is necessary to revisit the original logic for the Capability Maturity Model® (CMM®) and Capability Maturity Model® Integration (CMMI)® work [5, 6, 7]. The original framework used by Software Engineering Institute's (SEI) Software Engineering Process Management (SEPM) Program to characterize its process-improvement mission can be explained in terms of *what*, *how*, and *why* perspectives as shown in Figure 1. The *why* in this figure describes the need to respond to customer, acquirer, user, or management demands for *better* software engineering practices and results. The *what* and *how* define SEPM's approach for addressing these needs.

Here, a specific *why* question might be the following: Why would I need to improve my process? The answer might be the following: Because, by following your current process, your organization consistently misses most commitments to its customers, acquirers, users, and management. A *what* question might then be the following: What can I do to get my organization to consistently meet its commitments? Part of the answer to this question might be the following: Require your organization to plan all of its development work. Then a *how* question could be the following: How should we plan the development work? Finally, part of the answer to this question could be the following: Train the developers in sound planning methods, have them and their development teams plan their own work, and then have management review and negotiate these plans with the developers and their teams.

The objective of this article is to identify the relationships of the key process *whats* and *hows* and to suggest strategies to guide development organizations of all types in introducing superior engineering processes and in using these processes to consistently obtain superior performance. In doing this, we address two increasingly important issues: First, organizations often focus on maturity levels rather than process capability. Second, this focus can result in high maturity ratings that do not lead to better organizational performance. To address these two issues, organizations need specific guidance on *how* to do development work. By design, CMMI does not now provide guidance on *how* to do development work.

The CMM was SEPM's initial approach to addressing the *what* dimension. It initially focused on software engineering and was later broadened to encompass other engineering fields such as systems engineering and software acquisition. This work has been codified in the CMMI model. Other aspects of the *what* dimension that are not discussed here are appraisal, measurement, and analysis.

Figure 1: *SEPM's Strategic Framework*



SEPM addressed the *how* dimension of software engineering with the Personal Software Process^{SM} (PSP^{SM}) and the Team Software Process^{SM} (TSP)^{SM}. When developers in CMMI-assessed organizations use the PSP and TSP, the assessors can use the data generated by the PSP and TSP to verify the proper use of mature development processes. The PSP and TSP are now being adapted for general use by acquisition, systems-engineering, and hardware and software teams.

It is easy to become confused about what is a *what* and what is a *how* in process improvement and how these aspects relate and fit together. Before addressing this issue, however, it is important to first describe what a superior process is and outline some of the current issues facing the process improvement community in general and the CMMI community in particular.

A Superior Process

The engineering processes of the future must meet the following five requirements:

1. Control development costs and schedules predictably.
2. Respond to changing needs.
3. Minimize development costs and schedules.
4. Be scalable from small to very large systems.
5. Produce quality products predictably.

These topics are covered in the following sections.

Control Development Costs and Schedules

Cost and schedule problems are not new, and many other fields have learned how to manage them. For knowledge work, the solution has always been the same:

- The people who will do the work estimate and plan that work.
- Sound methods and relevant data are used to plan, track, and manage the work.
- The progress of the work is precisely and regularly tracked.
- When progress falls behind plan, the problem causes are promptly identified and resolved.
- When the requirements change, all involved levels promptly re-estimate and revise the entire plan.
- Risks are anticipated and managed.

While one might question the applicability of this approach to systems, software, and hardware development, there is now ample evidence to demonstrate its efficacy. Nearly 20 years of experience with process improvement have shown that these principles, when properly applied at all levels, can have important benefits [8, 9, 10, 11].

Respond to Changing Needs

While responsively handling changing needs would seem like a simple issue, it is not. The problem is to be responsive to new information while continuing to meet prior cost and schedule commitments. In fact, it is just this trade-off that is responsible for many of the severe cost and schedule problems of large systems. To be responsive but still maintain project control, development groups must do the following:

- Examine every proposed change to understand its effects on the development plan.
- Pay particular attention to each change's impact on completed work, including the requirements, design, implementation, verification, and testing activities.
- Estimate all cost and schedule consequences of making the needed changes.
- Where the cost and schedule implications are significant or where they exceed the currently approved plan, get management approval before proceeding.

Minimize Development Costs and Schedules

The three most common ways to minimize development costs and schedules are the following:

1. Optimize project staffing.
2. Reduce the amount of work.
3. Minimize the amount of rework.

While the actions required to address points one and two are reasonably straightforward and do not need further discussion here, point three is not and is discussed later.

Be Scalable

A scalable process must follow principles and practices that are suitable for the sizes of the projects with which it will be used. To be scalable, a process must meet the following three criteria:

1. It must use robust and precise methods at all levels, especially at the working systems-, software-, and hardware-engineer levels.
2. For technical and management program decisions, the management system must be based on and give great weight to the knowledge and judgment of the development-level professionals and anyone else who has relevant information.
3. The process must consistently use data that are derived from accurate, precise, and auditable process and product measurements.

Produce Quality Products

The governing quality consideration for large-scale systems development is that a high-quality process will consistently produce high-quality products while a poor-quality process will generally produce low-quality products. The problem here is with the word *generally*. People tend to remember their occasional successes and forget their less memorable achievements. As a result, when a development group has produced a seemingly high-quality, small product with an unmeasured and poorly controlled process, the members tend to feel that they have proven that process and should continue to use it, even for larger-scale work. However, unless they have measured and statistically verified that this process consistently produces quality products and that it is scalable, they run a significant risk of getting poor-quality results. With really massive monolithic systems, just a small chance of getting a poor-quality component would compound almost certain quality problems for the overall system.

Current Process Improvement Issues

While engineering groups face many challenges, the process management and improvement communities must now address two current issues. First, with increasing marketplace pressure, organizations often focus on maturity levels rather than process capability. Maturity levels cannot comprehensively measure organizational capability. They can indicate management priorities and the degree to which an organization is attempting to address its process problems. They can also guide the search for risky process areas and help establish process improvement priorities.

We now see cases where high-maturity ratings do not always result in the rated processes being used on the subsequent projects. It is not that the appraisal process is faulty or that organizations are dishonest, merely that the maturity framework does not focus on *how* the work is actually done; it only addresses *what* is done. While this can be adequate when organizations are truly striving to achieve a superior process, a concentration on

^{SM} Team Software Process, Personal Software Process, TSP, and PSP are service marks of Carnegie Mellon University.

maturity levels can cause organizations to ignore the *how* aspect of process improvement and adopt inefficient and poor-quality methods and practices.

The second issue concerns adjusting the CMMI framework and assessment methods to address this maturity-level problem. Without change, we can expect more cases where high-maturity ratings do not consistently lead to better organizational performance. Three lessons from experience suggest ways that help to ensure improved maturity levels consistently lead to improved organizational performance:

1. To properly control complex and precise work, everyone must manage to detailed and precise personal plans.
2. To predictably produce high-quality, large-scale systems, everyone must measure and manage quality.
3. The true measure of process improvement is then the degree to which the behavior of all of the working professionals and their teams reflects these practices.

To guide software developers in applying these lessons to their work, the SEI developed the PSP and the TSP and there is now substantial evidence that these methods can indicate the effective use of mature development processes [10, 12, 13]. The SEI is now adapting the PSP and TSP to general product-development work and investigating its use for acquisition work.

This, however, leads to a third issue: flexibility. CMMI does not define *how* to do development work. From the very beginning, the focus has been on *what* to do. The reason was that the original CMM was developed to guide the U.S. Department of Defense in source selection for software-intensive projects, and we did not want non-developers to specify how software development groups should do their work. While we had many ideas about how that process could be improved, we did not think that the users knew any better than industry how software should be developed. We also knew that, as current processes were improved, many new and creative methods were likely to be developed. By focusing exclusively on the *what* dimension, we hoped that CMM, and later CMMI, would not constrain process innovation. This continues to be our position.

Since we also believed that the *how* dimension of process improvement was important, we developed the PSP and TSP to provide guidance on how personal and team project planning, tracking, and quality management could be performed. Our concern now is to find ways to relate these practices and this guidance to the CMMI model without switching the focus from *what* to *how*. The need is to encompass both software and other development fields and not constrain development organizations as the technology advances. SEPM is working on these issues as we strive to improve the effectiveness of these methods.

Process Management Principles

Experiences with CMMI, PSP, and TSP have shown that, when an organization's management is convinced of the value of disciplined processes and properly implements an orderly and planned process improvement strategy, it can build the capability to successfully produce the truly large-scale systems of today and tomorrow. The five basic principles of these superior processes are as follows:

CMMI does not define how to do development work. From the very beginning, the focus has been on what to do *... By focusing exclusively on the* what *dimension, we hoped that CMM, and later CMMI, would not constrain process innovation.*

1. All modern science and engineering is based on learning from prior demonstrably effective practices. Competent engineers and scientists know what experiments have been successful and base their personal and team processes and practices on this experience. They stay current with new process developments and do not waste time experimenting with processes that have already produced unsatisfactory results. Until developers consistently use defined and proven processes, they will waste their time relearning known truths. Similarly, in experimenting with new and improved processes and methods, competent process professionals build on the results of prior experience.
2. With an inefficient or ill-defined process, developers must follow poorly defined and inaccurate plans. With the data available from a modern process, plans can be both accurate and precise. With defined and sound processes and precise and accurate plans, developers need not waste their time trying to find out what to do next and can devote more of their efforts to creative technical work.
3. Development is a learning process, and unless this learning is codified and preserved, the resulting knowledge is generally lost. That is the reason developers should define, use, and continually improve their processes: to build on their own and other's experiences.
4. For individuals and groups to work together effectively, they must coordinate their activities. While very small groups may be able to accomplish this informally without defined processes and detailed plans, large groups cannot.
5. Quality work is not done by accident or mistake. Quality must be planned, measured, tracked, and managed. When it is, product defect levels are normally reduced by orders of magnitude [10]. With current commonly used practices, large software products typically have thousands of test defects and developers spend at least half of their time finding and fixing enough defects for the product to run the basic tests. Even then, finished products generally have many unidentified defects. While it takes considerable skill to find and fix defects in test, fixing defects is not creative work. High-quality work is only produced by people who strive to produce quality products with every step of their work.

To have a reasonable chance of being successful, the more challenging engineering programs of the future must address the critical systems-development problems of cost and schedule, requirements instability, process management, and quality management. In fact, the use of defined, planned, measured, and quality-controlled processes can help improve both the business and technical performance of large-scale programs. Later sections of this article describe the actions required to accomplish this. First, however, it is important to define the principles of quality management.

Quality-Management Principles

Newer systems-of-systems structures are now being considered for many programs because they offer many advantages. As demonstrated by the Internet, the best-known system-of-systems, such structures can be improved by many groups independently and they can generally withstand individual node failures without disabling the entire system. However, they also have disadvantages. For example, their high reliability and relative node independence requires substantial redundancy and, thus, higher costs for the individual system nodes. Such structures must also have relatively narrow bandwidth coupling among the nodes. This can severely restrict system performance.

For these reasons, most large complex systems are built as small collections of large monolithic nodes instead of as large collections of small independent nodes. This is because of the potential for increased performance, security, economy, or some other key property that could not be obtained by decoupling the system into relatively independent, small elements. The tight-coupling characteristics of large-scale systems generally result from optimizing the overall design and from minimizing redundancies and inefficiencies among the system's component parts. This results in closer coupling among the system's components and large numbers of critical interdependencies.

To produce superior systems and especially the types of large-scale systems needed in the future, all aspects of process quality must be addressed. This includes the business aspects of quality management such as cost, schedule, predictability, and risk management. It also includes the marketing aspects of quality like competitive superiority and customer satisfaction. Finally, particularly for large-scale systems, the process used must be effective for the scale of the work being undertaken. The quality methods required for a process to scale up to handle the kinds of large-scale systems-development work that we can expect in the future are based on the following four critical assertions:

1. To have any hope of producing quality work, the overriding process goal must be to prevent defects of all kinds before they are introduced into the system.
2. To produce quality products consistently, the process must also have the objective of removing all defects before test entry. The objective of testing then becomes to verify and validate the product – not to fix its defects.
3. To ensure an effective and high-quality process, that process must provide quality measures.
4. To manage a quality process effectively, everyone from the senior executives to the individual systems-, hardware-, and software-engineering team members must support and participate in that quality process.

These assertions are briefly discussed in the following sections.

Preventing Defects

The software- and systems-development communities have long focused on defect prevention, but they have not generally approached it as a quality-management activity. For example, efforts to improve requirements-development work are defect-prevention activities, as is the adoption of better design methods or the use of improved implementation or design tools. For these efforts to be most beneficial, however, they should include explicit defect and productivity measures and analyses to ensure that they are addressing the most critical defect sources in the most effective way.

Removing Defects Before Test

Removing defects before test is accepted as an essential element of all quality-management programs. By following traditional quality-management principles, organizations minimize rework, reduce project costs and schedules, and produce better products. The following principles of quality management are based on facts that have been demonstrated in every field where they have been tested, including software [13, 14, 15]:

- It costs more and takes longer to build and fix defective products than it would have taken to build them properly the first time.
- It costs more to fix a defective product after delivery to users than it would have cost to fix it before delivery.
- It costs more and takes longer to fix a product in the later testing stages than in the earlier design and development stages.
- It costs more and takes longer to fix requirements and specification errors in the design, implementation, test, and operational stages than in the earlier requirements and specification stages.
- It is least expensive and most efficient to prevent the defects altogether.

These quality principles are based on experiences with both small- and large-scale systems of all types. The only point of debate typically concerns requirements defects. Here, however, the issue is not really one of correcting known defects as much as with understanding the system's true requirements and how to achieve them. Where the requirements are not clear, it is often best to make a first cut at the requirements and then test the related functions in practice. Whenever the requirements for new or highly modified systems or platforms are known to be wrong, it is always cheapest to fix them at the earliest possible point in the process. However, if the requirements for a fielded system or platform are later found to be incorrect, the requirements and possibly even the entire system strategy may have to be re-evaluated.

Even though the above quality principles have been proven in every case where they have been properly applied, they are still not universally accepted. The reason is that engineering organizations often establish separate groups for product development, production, testing, and field repair. Therefore, while the total operation would save time and money by following sound quality practices, the added costs of quality work must be borne by some groups while the savings accrue to others. Since quality programs typically increase costs in the early program stages and reduce them in later phases, the early investments in quality programs are generally hard to justify, particularly when organizations do not have the quality data to support their introduction. Finally, unless managers and developers have personally experienced the benefits of an effective quality program, few are willing to make the necessary effort to do high-quality work.

Quality Measures Are Essential

Today's commonly used systems-, hardware-, and software-development processes do not incorporate quality measurement and analysis at the individual and team level. This is a crucial failing since with even rudimentary measures, the above-named facts would be obvious and, as a consequence, more efficient and sensible quality practices would have been adopted long ago. The simple fact is that without precise quality measurement and analysis at all working levels, no serious quality program can be effective. While developers can make rudimentary quality improvements without measures, achieving the defect levels required in modern complex systems must involve defect levels of a very few parts per million. Such levels are not achievable without complete, consistent, precise, and statistically based quality measurement and analysis.

Everyone Must Participate in the Quality Program

The requirement that everyone must participate in the quality program is a direct consequence of the previously stated facts. To achieve defect levels of a few parts per million, quality must be a high priority for everyone. Quality work results only from a consistent striving for perfection. The individual development team members must all strive to produce defect-free work. Any undisciplined work by any acquisition professional, systems engineer, hardware developer, software developer, tester, or almost anyone working on or with the product can be a source of defects, and their work must be measured and quality controlled. If it is not, high-quality products simply will not be produced.

This is not only true for all of the development team members; it is also true for all of the support groups, managers, and executives. For example, an executive decision to skip some review, test, or check because *we don't have the time* will inevitably lead to poor-quality products. The only acceptable attitude at all levels of an entire program and in all involved engineering organizations must be *we don't have time to do it wrong!*

The CMMI and TSP Direction

The characteristics of superior engineering and management processes have now been clearly demonstrated in many fields, but it has become clear that to establish such processes, many engineering organizations need more precise *how-to* guidance than that provided by CMMI. For this reason, SEPM is exploring ways to provide such guidance. While the initial SEPM efforts in this direction were to establish a separate effort called the TSP, additional steps are needed, such as the following:

1. The TSP, which was initially focused on software teams, is now being broadened with a TSP Integrated (TSPI) project to include all types of engineering teams.
2. Because CMMI and TSP follow the same process- and quality-management principles, their joint use reduces TSP introduction costs and accelerates CMMI improvement. This has been demonstrated by the greatly accelerated process improvement schedules achieved by organizations that coordinate the use of both methods [11].
3. Work-to-date has demonstrated that there are substantial synergies between the CMMI and TSP methods. For example, an SEI project has mapped the TSP practices onto the CMMI framework and identified alignment gaps and overlaps [16]. Further, such joint work among the developers and users of the CMMI and TSP is planned.
4. Based on the work completed to date, the SEI will couple the CMMI and TSPI work more closely to provide a more coherent improvement road map for the process improvement community.

Conclusion

The current commonly used systems development methods have reached (or soon will reach) their feasibility limits, and continuing to develop the increasingly challenging and massive systems of the future with the most widely used methods of today is destined to failure. The danger is that with the rapid pace of technology, society could well be lulled into the false belief that the technical community is capable of building the systems we can technically describe. As these newer systems are used to support increasingly critical aspects of modern society, we then will likely face far more catastrophic system failures than we have experienced previously.

To successfully produce the large complex and critical systems of the future, development organizations must start to use more modern and consistently successful processes. These processes must follow sound management and quality principles. In particular, the people doing the work must plan their own work and that work must be precisely and continuously tracked. Everyone in the organization must be involved in and completely committed to the organization's quality management program, and management must recognize and reward superior work.◆

Acknowledgements

We thank the reviewers who made many helpful suggestions and comments on this paper. They are Bob Cannon, David Carrington, Caroline Graettinger, Jim McHale, Julia Mullaney, Dennis Smith, and Dave Zubrow. We also thank the CROSSTALK staff and reviewers for their help in reviewing, refining, and producing this article.

References

1. Neumann, Peter G. Computer Related Risks. Reading, MA: Addison Wesley, 1995.
2. Petroski, Henry. To Engineer Is Human: The Role of Failure in Successful Design. New York: Random House (Vantage Books), 1992.
3. United States. General Accounting Office (GAO). Observations on FAA's Air Traffic Control Modernization Program <www.gao.gov/archive/1999/r199137t.pdf>.
4. Gross, Grant. "FBI Trying to Salvage $170M Software Package." Computerworld 14 Jan. 2005 <www.computerworld.com/databasetopics/data/story/0,10801,98980,00.html>.
5. Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. CMMI 2nd Edition - Guidelines for Process Integration and Process Improvement. Reading, MA: Addison-Wesley, 2007.
6. Humphrey, W.S. Managing the Software Process. Reading, MA: Addison-Wesley, 1989.
7. Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, MA: Addison Wesley, 1995.
8. Gibson, Diane L., Dennis R. Goldenson, and Keith Kost. "Performance Results of CMMI-Based Process Improvement." CMU/SEI-2006-TR-004, ESC-TR-2006-004. Pittsburgh, PA: SEI, Carnegie Mellon University (CMU), 2006.
9. Humphrey, W.S. Winning With Software: an Executive Strategy. Reading, MA: Addison-Wesley, 2002.
10. Davis, N., and J. Mullaney. "Team Software Process in Practice." SEI Technical Report CMU/SEI-2003-TR-014, ESC-TR-2003-014. Pittsburgh, PA: SEI, CMU, 2003.
11. Pracchia, Lisa, "The AV-8B Team Learns Synergy of EVM and TSP Accelerates Software Process Improvement." CROSSTALK Jan. 2004 <www.stsc.hill.af.mil/crosstalk/2004/01/index.html>.
12. Humphrey, W.S. PSP: A Self-Improvement Process for Software Engineers. Reading, MA: Addison-Wesley, 2005.
13. Humphrey, W.S. TSP: Coaching Development Teams. Reading, MA: Addison-Wesley, 2006.
14. Deming, Edwards W. "Out of the Crisis." Cambridge, MA: MIT Center for Advanced Engineering Study, 1982.
15. Juran, J.M., and Frank M. Gryna. "Juran's Quality Control Handbook." 4th ed. New York: McGraw-Hill Book Company, 1988.
16. McHale, James, and D. Wall. "Mapping TSP to CMMI." CMU/SEI-2005-TR-014, ESC-TR-2005-014. Pittsburgh, PA: SEI, CMU, Apr. 2005.

About the Authors

Watts S. Humphrey joined the SEI after his retirement from IBM in 1986. He established the SEI's Process Program and led development of the SW-CMM, the PSP, and the TSP. During his 27 years with IBM, he managed all IBM's commercial software development and was Vice President of Technical Development. He holds graduate degrees in physics and business administration. He is an SEI Fellow, an Association of Computing Machinery member, an Institute of Electrical and Electronics Engineers Fellow, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. In a recent White House ceremony, the President awarded him the National Medal of Technology.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-6379
Fax: (412) 268-5758
E-mail: watts@sei.cmu.edu

James W. Over is a senior member of the technical staff at the SEI. Since joining the SEI in 1987, he has worked in several technical areas, including TSP, PSP, Software Process Definition, and Software Process Modeling. His interests include software engineering, software process, and quality management. Over co-authored several SEI publications on software process definition and improvement. Before joining the SEI, he was director of information systems for Pulitzer Publications. Over has more than 34 years of technical and management experience in software engineering, and attended Northern Illinois University.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-7624
Fax: (412) 268-5758
E-mail: jwo@sei.cmu.edu

Michael C. Konrad, Ph.D., is chairman of the CMMI Configuration Control Board and has been a team leader of CMMI model development since 1998. He was also a member of teams that developed Software CMM Vers. 1.0, Software Development Capability Evaluation, and International Organization for Standardization 15504 model requirements. Konrad has 24 years experience in software engineering, holding various positions in industry and academia. He received his doctorate in mathematics from Ohio University in 1978.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-5813
Fax: (412) 268-5758
E-mail: mdk@sei.cmu.edu

William C. Peterson worked at IBM for more than 23 years, and has been with the SEI for 13 years. At IBM, he managed software development projects and led process improvement efforts. At the SEI, he is director of the SEPM program where he is responsible for the CMMI, the TSP, the Software Engineering Measurement and Analysis, and the International Process Research Consortium initiatives.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-7736
Fax: (412) 268-5758
E-mail: wcp@sei.cmu.edu

Coming Events

March 12-15
23rd Annual National Test and Evaluation Conference
Hilton Head Island, SC
www.ndia.org

March 14-15
SQuAD 2007 6th Annual Conference Software Quality Association of Denver
Denver, CO
www.geocities.com/lbu_measure/conf.htm#p2

March 21-23
The ServerSide Java Symposium
Las Vegas, NV
http://javasymposium.techtarget.com/lasvegas/index.html

March 25-29
CNS '07
10th Communications and Networking Simulation Symposium
Norfolk, VA
www.scs.org/confernc/springsim/springsim07/cfp/cns.htm

March 25-29
SpringSim '07
Norfolk, VA
www.scs.org/confernc/springsim/springsim07/springsim07.htm

March 26-29
SEPG 2007
Austin, TX
www.sei.cmu.edu/sepg/2007

June 18-21, 2007
2007 Systems and Software Technology Conference

Tampa Bay, FL
www.sstc-online.org

Coming Events: **Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. E-mail announcements to nicole.kentta@hill.af.mil.**