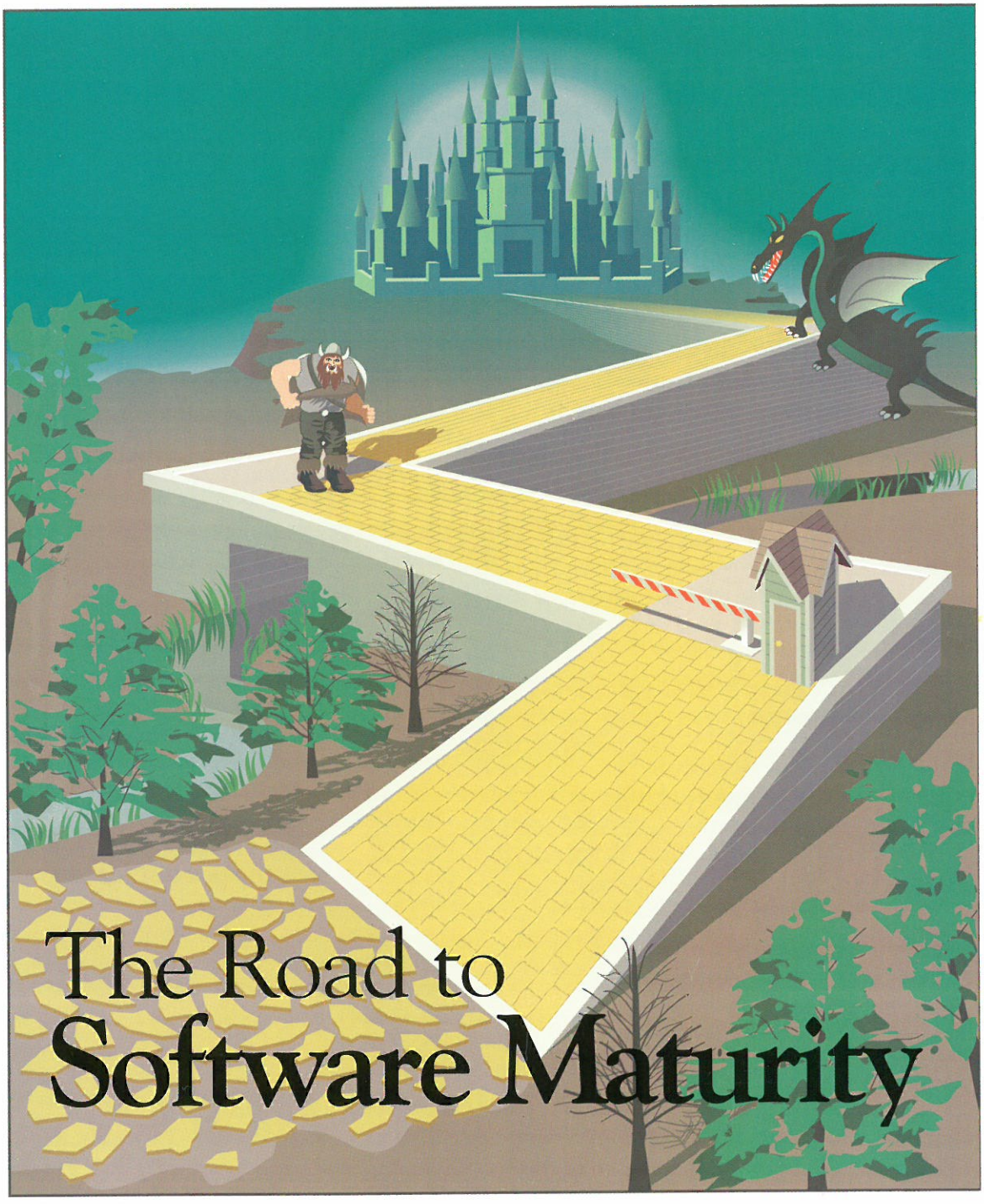
JANUARY 1995

# COMPUTER

*Innovative technology for computer professionals*

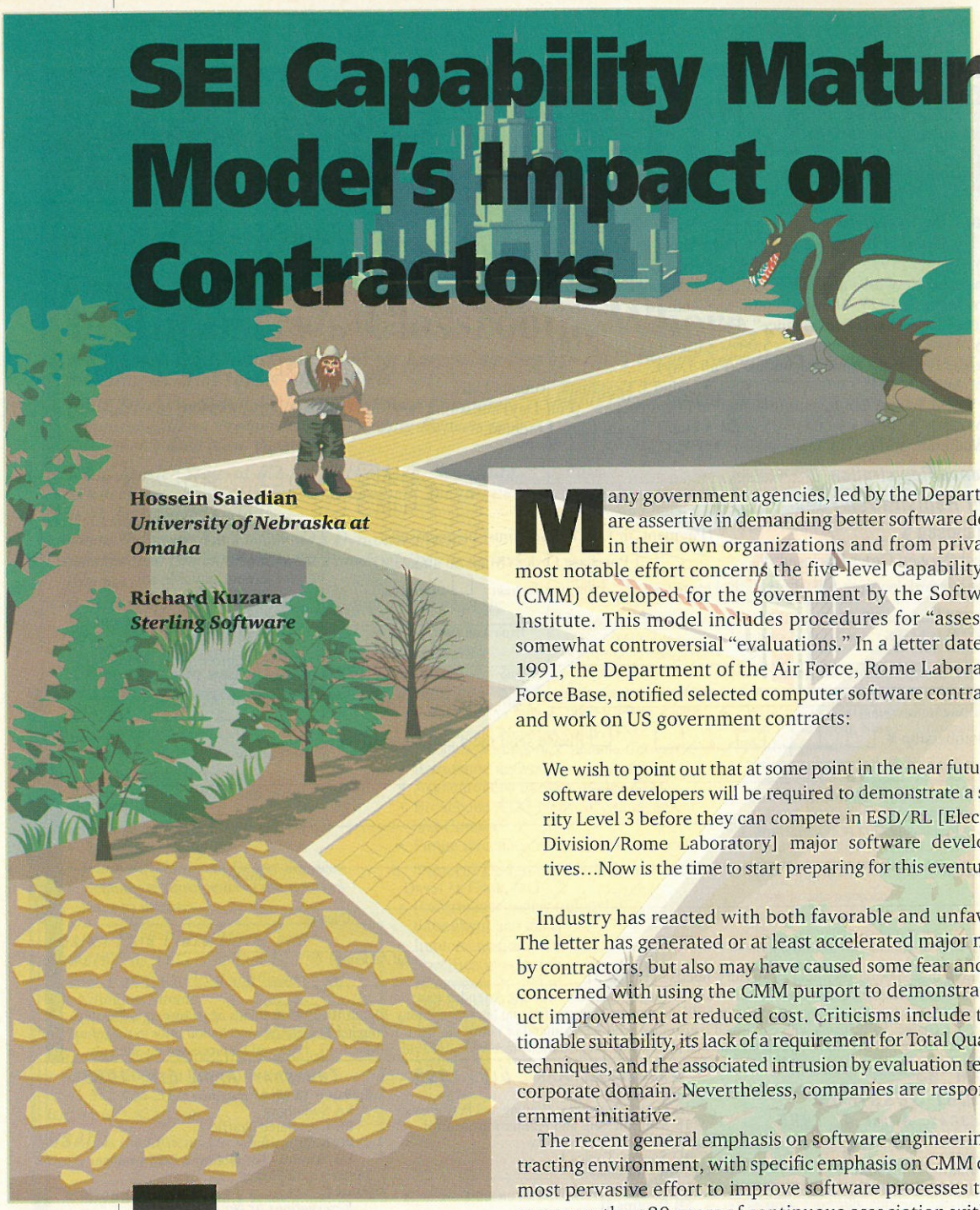


# The Road to Software Maturity

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

# SEI Capability Maturity Model's Impact on Contractors



**Hossein Saiedian**
*University of Nebraska at Omaha*

**Richard Kuzara**
*Sterling Software*

**With strong DoD sponsorship, more companies will probably base their software process improvement efforts on SEI's Capability Maturity Model, but the opposition is loud and clear.**

Many government agencies, led by the Department of Defense, are assertive in demanding better software development within their own organizations and from private industry. The most notable effort concerns the five-level Capability Maturity Model (CMM) developed for the government by the Software Engineering Institute. This model includes procedures for "assessments" and the somewhat controversial "evaluations." In a letter dated September 25, 1991, the Department of the Air Force, Rome Laboratory, Griffiss Air Force Base, notified selected computer software contractors who bid for and work on US government contracts:

> We wish to point out that at some point in the near future, all potential software developers will be required to demonstrate a software maturity Level 3 before they can compete in ESD/RL [Electronic Systems Division/Rome Laboratory] major software development initiatives...Now is the time to start preparing for this eventuality.

Industry has reacted with both favorable and unfavorable opinions. The letter has generated or at least accelerated major new undertakings by contractors, but also may have caused some fear and turmoil. Studies concerned with using the CMM purport to demonstrate software product improvement at reduced cost. Criticisms include the model's questionable suitability, its lack of a requirement for Total Quality Management techniques, and the associated intrusion by evaluation teams in the private corporate domain. Nevertheless, companies are responding to this government initiative.

The recent general emphasis on software engineering within the contracting environment, with specific emphasis on CMM compliance, is the most pervasive effort to improve software processes that we've seen in our more than 30 years of continuous association with software development. Only time will tell whether this undertaking will actually produce major positive results or whether its current high visibility will be allowed to gradually fade. But undoubtedly the government *can* cause major changes in the software contracting industry and, if it continues with the spirit and intent of the above quote, probably will.

## SOFTWARE PROCESS MATURITY

Software Process Maturity is a model developed by the Software Engineering Institute (SEI) at Carnegie Mellon University. This model attempts to quantify a software organization's capability to consistently and predictably produce high-quality software products.

Historically, software efforts emphasized products such as operating systems and new languages or techniques such as transaction processing.



0018-9162/95/$4.00 © 1995 IEEE

Today, software pervades all aspects of life, but the "expert" consensus is that software development methods are generally very poor. Software engineering has emerged to bring engineering principles and discipline to what has traditionally been an art or craft.

The US government—more specifically, the Department of Defense (DoD)—has always been a major software purchaser and has contended with poor software, missed schedules, and high costs. An unpublished review of 17 major DoD software contracts found that the average 28-month schedule was missed by 20 months, one four-year project was not delivered for seven years, and no project was on time.[1] In 1982, the DoD formed a joint-service task force to analyze its software problems. Initiatives included establishing the SEI and developing the well-known Ada Program. But recent years have seen an order-of-magnitude growth in software size and complexity, making it impossible to upgrade current software techniques without a fundamental process change.

In 1984, the SEI was established to address the DoD's need for improved software. Data collected by the SEI indicated that most US software-development organizations do not possess or use a defined, shared development model.[2] As a result, the Software Process Maturity Model was developed for DoD and industrial-software organizations. The Air Force asked the Mitre Corporation to participate in this effort, and the SEI-Mitre team produced a questionnaire and framework for evaluating organizations on the maturity of their software processes. This effort combined previous industry work with W. Edward Deming's principles and Walter A. Shewhart's process management concepts (described in Deming's book, *Out of Crisis*).

In 1991, the SEI produced the Capability Maturity Model. The CMM serves as a framework to continuously evolve and improve the related SEI questionnaire. A Questionnaire Advisory Board has been established to review SEI work and determine whether proposed changes to this work are suitable. To balance the needs and interests of those most affected, this board includes both US industry and government members.

## Capability Maturity Model

The CMM is a five-level model (see Figure 1).[3] The model is designed so that capabilities at lower stages provide progressively stronger foundations for higher stages. Each development stage—or "maturity level"—distinguishes an organization's software process capability.

The CMM and the associated questionnaire have two major uses: assessments and evaluations.[1] With assessments, organizations use the maturity model to study their own operations and identify the highest priority areas for improvement. Results form the basis for an organization's self-improvement action plan. Acquisition agencies use the maturity model to identify qualified bidders and monitor existing contracts. Results help develop a risk profile that augments the traditional criteria used to select the most responsive and capable vendors.

**CMM LEVELS.** For easy reference, the five CMM levels have been abbreviated as initial, repeatable, defined, managed, and optimizing. These levels have been selected by the SEI because they[1]

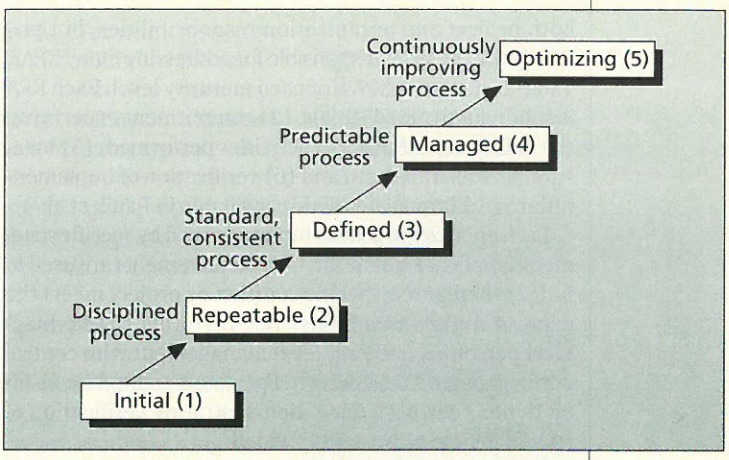



**Figure 1. Capability Maturity Model levels.**

- reasonably represent historical phases of evolutionary improvement,
- provide achievable improvement steps in reasonable sequence,
- suggest interim improvement goals and progress measures, and
- provide immediate improvement priorities once an organization's status in this framework is known.

Generally, the levels are characterized and distinguished as

1. *Initial*: While there are many degrees of management control, the first step is to roughly predict schedules and costs. This level has been described with many different catchy phrases such as "ad hoc," and "chaotic." Bollinger and McGowan[4] refer to it as "really not even a level at all, but the logical equivalent of an F—a failing grade." Until the process is under management control, orderly progress in process improvement is not possible.
2. *Repeatable*: The organization has achieved a stable process with a repeatable management control level by initiating rigorous project management of commitments, costs, schedules, and changes.
3. *Defined*: The organization has defined the process as a basis for consistent implementation and better understanding. At this point, the risk of introducing advanced technology is greatly reduced.
4. *Managed*: The organization has initiated comprehensive process measurements and analysis. This is when the most significant quality improvements begin.
5. *Optimizing*: The organization now has a foundation for continuously improving and optimizing the process.

**KEY PROCESS AREAS.** Each CMM level except Level 1 includes key process areas (KPAs) that identify where an organization must focus to raise software processes to that level. (Because KPAs are the requirements for achieving a maturity level, no KPAs are defined for achieving Level 1.) When an organization collectively performs the activities defined by KPAs, it can achieve goals considered important for enhancing process capability. All KPAs include

both project and organization responsibilities, but primarily the project is responsible for addressing many KPAs. Table 1 shows the KPAs for each maturity level. Each KPA is subdivided into (1) goals, (2) commitment to perform, (3) ability to perform, (4) activities performed, (5) measurement and analysis, and (6) verification of implementation. (Additional details are available in Paulk et al.[3])

Each area except goals is further defined by specific statements applicable to the area. These statements are used to judge whether the specific contract or project meets the expressed criteria. Judgments are performed by working-level personnel and first-level management who control performance on the contract. For statements to be considered met, "hard evidence" demonstrating verification of statement intent must be provided.

**Table 1. Key process areas by maturity level.**

| | |
|---|---|
| Level 5: Optimizing | Defect prevention<br>Technology-change management<br>Process-change management |
| Level 4: Managed | Quantitative process management<br>Software quality management |
| Level 3: Defined | Organization process focus<br>Organization process definition<br>Training program<br>Integrated-software management<br>Software product engineering<br>Intergroup coordination<br>Peer reviews |
| Level 2: Repeatable | Requirements management<br>Software project planning<br>Software project tracking and oversight<br>Software subcontract management<br>Software quality assurance<br>Software configuration management |

Basically, hard evidence refers to a philosophy statement about how something is done and evidence that the philosophy is consistently performed. An evaluation team will require hard evidence when judging the CMM level of a company or contract. What appears to be hard evidence to some, may not be to others. For example, a project may use a programmer notebook to specify exactly how a code inspection must be performed, but this alone is probably insufficient as hard evidence. In addition, a report for each inspected code module should specify a check or verification block for each code inspection step and should contain verification-personnel signatures along with the dates that steps were performed.

This stage's end result is to identify the specific statements within each KPA that are or are not currently being accomplished. For those being accomplished, hard evidence is identified and collected. This serves to guide the next process improvement stage: developing action plans to address process deficiencies.

## Assessments

A software process assessment is initiated by an organization to help improve its software development practices. The assessment is generally conducted by six to eight of the organization's senior software-development professionals and by one or two coaches from the SEI or from an SEI-licensed assessment vendor. The assessment is typically conducted in six phases:[1]

1. In the selection phase, the organization is identified as an assessment candidate, and the qualified assessing organization conducts an executive-level briefing.
2. In the commitment phase, the organization commits to the full assessment process when a senior executive signs an assessment agreement.
3. In the preparation phase, the organization's assessment team receives training, and the on-site assessment process is fully planned. All assessment participants are identified and briefed. The maturity questionnaire is filled out at this time.
4. In the assessment phase, the on-site assessment is conducted in about one week. Then the assessment team meets to formulate preliminary recommendations.
5. In the report phase, the entire assessment team helps prepare the final report and present it to assessment participants and senior management. The report includes team findings and recommendations for actions.
6. In the assessment follow-up phase, the assessed organization's team, with guidance from the assessment organization, formulates an action plan. After approximately 18 months, the organization should do a reassessment to assess progress and sustain the software process improvement cycle.

The assessment represents a major resource commitment by the organization and can be accomplished only with senior management's honest commitment and involvement. The assessing organization treats the assessment and its results as confidential information. The organization being assessed controls the assessment information and its exposure.

## Evaluations

In contrast to the voluntary, confidential assessment process (described above), a software capability evaluation (SCE) is typically conducted by an outside organization such as the government or a software contractor. It is intended to help the acquisition agency understand the management and engineering processes used by a bidder. It is assumed that the bidder will submit to the SCE process if it wishes to win the contract on which it bid.

Organizations that are candidates for an SCE first complete a maturity questionnaire. An evaluation team visits the organization and uses the maturity questionnaire to help select representative practices for a detailed examination. This examination generally consists of interviewing the organization's personnel and reviewing the organization's software-development-related documentation. By investigating processes used in the organization's current projects, the team can highlight specific records for review and quickly identify potential risk areas. Potential risk categories[1] considered include the likelihood that

- the proposed processes will meet the acquisition needs,
- the organization will actually install the proposed processes, and
- the organization will effectively implement the proposed processes.

The SCE is a very judgmental process, and it is mandatory that all organizations for a single contract be evaluated consistently. The SEI believes that the SCE method provides the needed consistent criteria and method.

Evaluations require significant time, travel, and other resources. The acquisition agency and its evaluation team spend many weeks preparing for and performing evaluations. The organization being evaluated is significantly impacted if it is interested in a successful evaluation. It will expend much effort emphasizing its strengths. Its personnel will be warned and briefed. Documentation and files will be organized and made ready. During the evaluation, interviewed personnel will be under great pressure to respond honestly but with properly chosen words. Some may feel that if they answer incorrectly, they'll be responsible for losing the contract. This is not to imply that organizations intentionally deceive evaluation teams, but the organization and its personnel can be expected to present the most favorable image possible.

### What the CMM does not address

The CMM is based on the premise that major software-development problems and, hence, causes for software project failures are managerial rather than technical. The CMM applies process management and quality-improvement techniques to software development and maintenance and therefore models organizational process improvement. The CMM, however, is not an exhaustive model or "silver bullet." It does not address several software management and engineering practices important for successful projects. For example, the CMM does not yet directly address expertise in a particular application domain; advocate specific tools, methods, or software technologies; or address issues related to human resources (such as how to select, hire, motivate, and retain competent people). Neither does it address issues related to concurrent engineering, teamwork, change management, or systems engineering. The authors of the CMM Version 1.1 acknowledge the above deficiencies in Paulk et al.[3]

There are other maturity models besides the CMM. One notable example is Capers Jones's model, considered by some as comparable or even superior to the CMM and purported to be widely used in the commercial sector. The ISO standard 9001 specifies quality assurance guidelines for software-system design, development, installation, and servicing. Since the CMM has its roots in government systems and defense-oriented software industry areas, it makes certain assumptions that may not be true in the commercial sector. This has prompted certain companies, such as Digital, to extend the CMM and make it applicable to their own process improvement efforts.

## INDUSTRY OPINIONS

As might be expected, opinions about the CMM and its associated assessments and evaluations include pro and con, with many shades of gray. Favorable opinions run from a lukewarm "it's better than nothing" to the rousing "achieving maturity saves millions." Unfavorable opinions typically cite individual deficiencies in the SEI model but do not appear to prove that it is worse than no model at all.

**THE CMM IS NOT AN EXHAUSTIVE MODEL OR "SILVER BULLET." IT DOES NOT ADDRESS SEVERAL SOFTWARE MANAGEMENT AND ENGINEERING PRACTICES IMPORTANT FOR SUCCESSFUL PROJECTS.**

### Favorable

One of the most detailed—and, apparently, carefully controlled—studies on implementing the SEI model concerns an assessment, actions, and reassessment at Hughes Aircraft's Software Engineering Division.[5] The SEI assessed six Hughes projects during November 9-12, 1987. The assessment team made seven recommendations on quantitative process management, process group, requirements, quality assurance, training, review process, and working relationship. (See sidebar on overleaf.)

The Hughes Software Engineering Division was assessed at Level 2 maturity. In early 1988, Hughes developed an action plan to implement recommended improvements that required 100 labor staff-months over an 18 month period. Budget cuts later reduced the labor staff-months to 78. In early 1989, Hughes requested that the SEI conduct a second assessment, which the SEI performed in 1990. Hughes had progressed to a strong Level 3, with many activities preparing it for Level 4 and 5.

Hughes has identified many advantages resulting from its improved software development processes. And perhaps more importantly, Hughes contends that improvement costs have been more than offset by a positive net return. Specifically, "The assessment itself cost Hughes about $45,000, and the subsequent two-year program of improvements cost about $400,000...Hughes estimates the resulting annual savings to be about $2 million."[5] (Humphrey, Snyder, and Willis[5] wholeheartedly favor the SEI model and its results and provide a deeper analysis of the above figures.)

Even though the CMM model is relatively new, there are other success stories. In early 1988, the Software Systems Lab at Raytheon's Equipment Division initiated a process improvement program.[6] An initial assessment, based on the SEI questionnaire, found that the lab was slightly below "repeatable" (Level 2) and four areas needed improvement: documented practices and procedures, training, tools and methods, and metrics.

In 1992, a follow-up analysis of six major Raytheon projects spanning three years showed substantially decreased rework costs since the start of the process improvement program. More specifically, Raytheon saved about $9.2 million of its nearly $115 million in software development costs. The approach chosen to quantify the software improvement initiative's effect was based on Phil Crosby's "cost of quality" idea (from his book, *Quality Without Tears*), which distinguishes the cost of doing something

## CASE STUDIES OF HUGHES AND RAYTHEON'S CMM EFFORTS

The software community has reached a stage where it has begun to formally define the software development process and the most effective ways to improve it. Several new "maturity" models have been proposed, but without quantitative evaluations it will be difficult to truly assess their impact. The SEI Capability Maturity Model is no exception. In fact, only a few quantitative studies have been conducted to evaluate the CMM's capability to improve the quality and predictability of software development, as well as its cost-effectiveness and return on investment. Following is a summary of two major case studies that offer concrete figures for CMM efforts at Hughes and Raytheon.

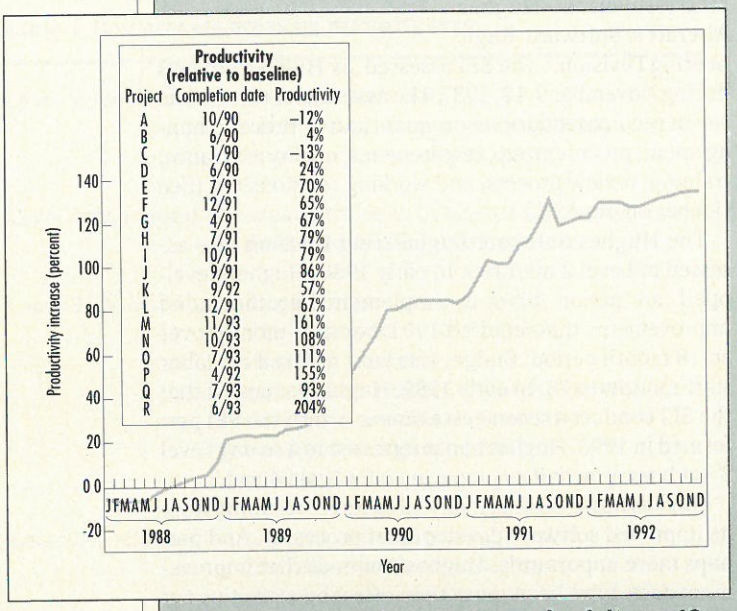



**Figure A. Average increase in productivity on 18 Raytheon projects (measured in equivalent delivered source instructions per person-month). Figure courtesy of *IEEE Software*.**

### What they accomplished

Hughes Aircraft's two-year program to raise its Software Engineering Division from level 2 to level 3 cost the company roughly $400,000 (75 person-months) from 1987-1990, a 2-percent increase in division overhead. This was allocated among six major functions:

- process-group leader (8 percent),
- process definition (6 percent),
- technology development (28 percent),
- quantitative process management (41 percent),
- training (16 percent), and
- review-process standardization (1 percent).[1]

Hughes calculated that its initial return on this investment amounted to $2 million annually based on a 50-percent reduction (from 0.94 to 0.97 percent) of its cost-performance index (budgeted cost of work performed/actual cost).[1-2] The business value of this investment was 4.5:1.[2] Hughes' CPI continued to improve through 1992, climbing from 0.97 to 1.02, to the point where projects as a whole were under budget.[2] Hughes attributes these savings to the new processes' early detection of defects, which substantially reduced rework costs. According to Herbsleb et al.,[2] savings at each stage for rework amounted to about

- 44 percent for preliminary design,
- 96 percent for detailed design,
- 83 percent for coding,
- 60 percent for unit tests, and
- 58 percent for integration tests.[2]

Raytheon's numbers are even more remarkable. Investing almost $1 million annually in process improvements, Raytheon achieved a 7.7:1 ROI (a $4.48 million return on $0.58 million) and 2:1 productivity gains (see Figure A).[3] Although allocation patterns change with perceived need, in 1992 funds were allocated as follows: policy and procedures, 18 percent; training, 23 percent; tools and methods, 30 percent; process database, 29 percent. Staffing is predominantly part-time and totals about 15 people per year, with one or two full-time personnel. Raytheon states that it has eliminated $15.8 million in rework costs (from 41 to 11 percent) on 15 projects tracked between 1988-1992 (see Figure B).

### How they accomplished it

These two case studies also lend substance to the types of process improvements deliberately left unspecified in CMM Version 1.1. Despite being tailored for specific business environments, the two process improvement plans share family resemblances based on the CMM model. Because the model requires senior management buy-in, the companies adopted a top-down approach, establishing what is commonly called a Software Engineering Process Group (SEPG) to develop, coordinate, and track the process improvement plans. The SEPG works to standardize policies and procedures, oversees the various technical working groups (TWGs) implementing process improvements, and provides a centralized organization-wide database for process-data analysis.[1] As reflected in the cost allocations, the two initiatives focused on three key areas.

#### *Quantitative process management*

Hughes standardized uniform data definitions across projects and used them to track cost estimates, actual costs, errors, and schedule performance. Information was compiled in a monthly report for senior management that included

the project's accomplishments, problems, program trouble reports, quality indicators, scope changes, resource needs, and lessons learned. Also presented were plots of actual versus planned values over time to show the project's schedule, milestones, rate chart, earned value, financial/labor status, and target-system resource use.[1]

Similarly, Raytheon's TWG for metrics issues adopted Mitre's Management Metric set for identifying "systemic problem areas in a development process" and created the Process Data Center to support "proposal writing, quarterly reviews, software-capability evaluations, and specific studies such as the predictive models necessary to achieve level 4 maturity."[3] The TWG also provided standardized spreadsheet templates to facilitate metrics collection by project members.

*Technology development*

Hughes' technology steering committee formalized technology management practices and procedures, and created a job function called head of technology transfer. Among other things, the head of technology transfer monitored process maturity, "maintained a database of technology used on each project and an awareness of what technology each project needed," and became involved in various corporate-wide programs relating to technology development, process maturity, and training.

Raytheon established a similar tools-and-methods working group that focused on evaluating tools and environments and on process automation. The program has

> sponsored the evaluation of alternative CASE products, the cost-benefit analyses used to justify their purchase, the training to instruct the developers in their intricacies, the integration of individual tools to provide a seamless environment, the inevitable tailoring to specific projects, and the generation of manuals for various types of users.[3]

*Training*

Both Hughes and Raytheon place heavy emphasis on training, with Hughes going so far as to make training a job requirement instead of a promotional requirement. In response to SEI recommendations and an employee survey, Hughes supplemented its classes on programming practices, languages, and CASE tools with classes on project management, internal reviews, requirements writing, requirements- and unit-level testing, and quality assurance.[1] Hughes opened these courses to engineering functions outside of its Software Engineering Division and reported a respectable non-Software Engineering Division attendance of 20 percent. Hughes believes that this enhanced training program contributes heavily to the "coherent organizational culture" achieved at level 3.[1]

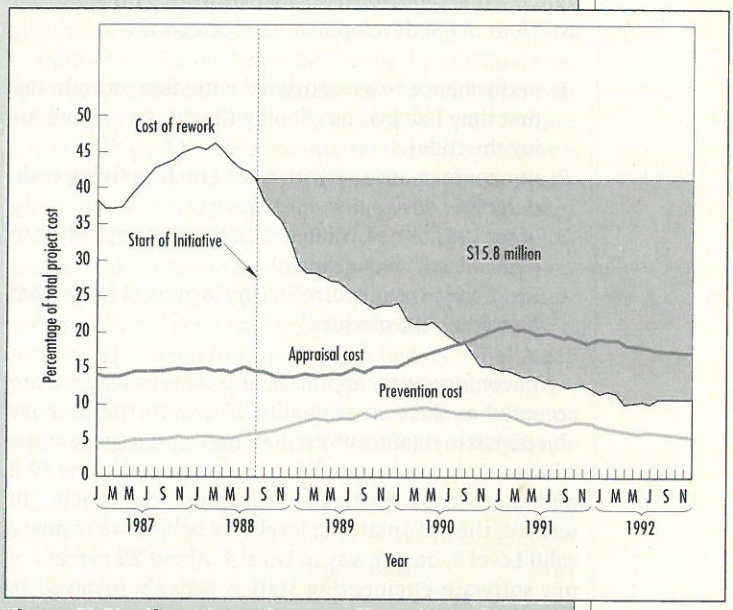



**Figure B. Raytheon's savings on 15 projects due to reduced rework costs. Figure courtesy of *IEEE Software*.**

Raytheon sponsors a comprehensive training program, with courses conducted during work hours (564 courses in 1992). Overview courses are designed to provide general knowledge about some technical or management area and are scheduled periodically. Detailed courses are often tailored for specific projects and scheduled accordingly. Like Hughes, Raytheon reports "a definite culture shift in the area of training."[3]

Although the ROI reported above (and elsewhere) suggests that CMM more than pays for itself, it can also be argued that corporations are more likely to report successful results than negative ones. To establish a more thorough understanding of the CMM's impact, we need a more extensive project-base (for example, 50 projects) to quantitatively, statistically, and comparatively report the results. It is probably too early in the evolution of the CMM to perform this type of study.

—Hossein Saiedian, University of Nebraska, and Scott Hamilton, *Computer* Staff

**References**


1. W. Humphries, T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software*, Vol. 8, No. 4, July 1991, pp. 11-23.
2. J. Herbsleb et al., "Benefits of CMM-Based Software Process Improvement: Initial Results," Tech. Report SEI-94-TR-13, Software Engineering Institute, Carnegie Mellon Univ., Pittsburgh, Aug. 94.
3. R. Dion, "Process Improvement and the Corporate Balance Sheet," *IEEE Software*, Vol. 10, No. 4, July 1993, pp. 28-35.

right the first time from the cost of rework. This approach used four major development-cost categories:

1. performance costs associated with doing it right the first time (such as developing the design or generating the code),
2. nonconformance rework costs (such as fixing code defects or design documentation),
3. appraisal costs associated with testing the product to determine if it's faulty, and
4. Prevention costs incurred trying to prevent faults from degrading the product.

Nonconformance, appraisal, and prevention costs are accepted as the cost of quality. However, the primary objective is to significantly reduce nonconformance costs, which clearly was accomplished (as witnessed by the $9.2 million savings). As stated in the referenced article, "In terms of the SEI's maturity levels, we believe we're now a solid Level 3, on our way to Level 4. About 25 percent of our software engineering staff is actively involved in process improvement, and our initiative has good visibility at all levels of management...."

**Table 2. Comparison between software process assessments and software capability evaluations.**

| Software process assessments | Software capability evaluations |
|---|---|
| Used by organization to improve software process. | Used by acquisition organization for source selection and contract monitoring. |
| Results to organization only. | Results to organization and acquirer. |
| Assess current practice. | Substantiate current practice. |
| Act as catalyst for process improvement. | Assess commitment to improve. |
| Provide input to improvement, action plan. | Analyze contract performance potential. |
| Collaborative: organization members on team. | Independent evaluation: no organization members on team. |
| Apply to overall organization, not individual. | Apply to performance for particular contract. |

As another example, the first Air Force center to perform a software engineering process self-assessment was the Aircraft Software Division (LAS) of the Oklahoma City Air Logistics Center at Tinker Air Force Base.[7] In the late 1980s, LAS struggled with the need to implement a strong, effective process improvement program, and in 1989 LAS was introduced to the SEI methods. The SEI helped assess LAS, which was challenged with developing a process-improvement infrastructure to correct the assessment findings and increase its organizational maturity. LAS established a management steering team and technical Software Engineering Process Group (SEPG).

By late 1992, LAS had implemented 44 improvements. Return-on-investment information gathered for 18 of these improvements reported $462,100 invested with $2,935,000 returned, for an ROI ratio of 6.35 to 1. In fact, LAS continued collecting such data, making a strong case for its process improvement program.

Further benefits include improved communications and accuracy. LAS officials believe that the improvement program has made process improvement a daily business priority. Employees actively seek improvement opportunities. Customer satisfaction, a primary LAS goal, has increased directly because of process improvement efforts. LAS is a leader in achieving the Air Force decree that all of its software organizations perform a self-assessment by 1994 and reach an SEI maturity Level 3 by 1998.

### Unfavorable

As might be expected, not all opinions about the SEI and CMM are favorable. Documented concerns address specific details of the CMM, assessments, and evaluations. There is a consensus regarding the need for software engineering improvement, but there is disagreement on specific issues.

The SCE, which represents intrusion into the contractor's previously "private" environment, generates controversy. A common theme is that this evaluation is being applied in many different ways and that the SCE method taught departs significantly from the one published.[8] The SCE method taught is based on CMM version 0, which identifies eight KPAs and has never been published. Although version 1 with 13 KPAs is the published version, the SEI apparently will wait for the next version before updating its SCE method (due in 1996).

At the Software Capability Evaluations Workshop (held July 16-17, 1992 in Pittsburgh), five speakers from government and Mitre described SCEs they had conducted or observed. No two speakers described the SCE method the same way (see Card[8] for many differences). Furthermore, even when the same method is applied, SCE results can vary greatly. In one case, two SCE teams evaluated the same organization only a month apart and got different results for 15 of 85 questions.

Common concerns expressed by five industry representatives during the aforementioned workshop included

- different SCE methods,
- questionable SCE team qualifications and training,
- SCE teams intimidating personnel,
- SCE teams not providing timely feedback,
- fuzzy compliance criteria,
- cost of supporting frequent SCEs, and

- discrepancies between Software Process Assessments (SPAs) and SCE results.

One Air Force representative observed that in seven of 14 cases, an SPA rated an organization higher than an evaluation did. This was attributed to misguided improvement efforts and poor contractor integrity. Government and industry representatives both contended that because SPAs often give different or misleading answers, they should be replaced by SCEs. But many industry representatives felt that because SPAs offered a more cooperative approach and covered more issues, they were useful for process improvement.

This thinking illustrates the common misconception that SCEs and SPAs must agree. Actually, they need not always yield the same rating. SCEs evaluate an organization's ability to perform the specific tasks required for fulfilling a contract; SPAs assess an organization's general maturity. An SCE result can legitimately differ from an SPA result if, for example, the organization bids on a contract outside its usual business sphere. As summarized in Table 2 (see Paulk[3] for details), SPAs and SCEs differ in motivations, objectives, and results ownership. These in turn lead to differences in the information collected and outcomes formulated. For instance, while SPAs are performed in an open, collaborative environment, SCEs are performed in a more audit-oriented environment, and objectives are tied to monetary considerations because team recommendations help select contractors.[3]

Pyzdek[9] presents arguments against the SEI and its somewhat rigid CMM. In this article, Pyzdek contends that there is no "right" way to improve software quality: Every organization must come up with its own approach. This implies that no single mandated approach is right, although the CMM comes close. This article also states, a quality improvement solution imposed from the outside is by definition not the answer. And the CMM is certainly from the outside. The counterargument is that the *methods* for achieving CMM levels can be chosen by the organization; it is only the *criteria* that are defined from the outside.

Another confusing major area has been the relationship between the CMM and Total Quality Management (TQM). Figure 2 shows the implied relationship between the CMM and TQM. According to this figure, while TQM principles may affect all of an organization's projects, the CMM affects only software development projects. Silver[10] claims there are several CMM flaws. He says the CMM

- ignores TQM and processes' cultural dimension,
- confuses processes' infrastructure and activity dimensions,
- institutionalizes quality assurance and process groups,
- poorly implements statistical process control,
- delays useful process improvement activities,
- doesn't account for parallel, interdependent, and continuous improvement of all KPA activities,
- provides no quantitative process-performance metrics, and
- ignores software support.

Silver[10] expands on each failing and argues that the CMM's major flaw is its failure to recognize TQM, a criticism stemming from the belief that TQM should serve as the foundation for quality.

Another article concludes that "while the SEI has developed a truly outstanding program for performing process assessments, both its assessment and the SCE program are seriously flawed by their reliance on the SEI's unproven process-maturity model. Furthermore, the methods by which the SCE program determines numeric process-maturity scores for organizations are so riddled with statistical and methodological problems that it appears unlikely that such ratings have any meaningful correlation to the actual abilities of organizations to produce high-quality software on time and within budget."[4] Bollinger and McGowan also criticize the process-maturity questionnaire of 101 yes/no questions on various software engineering-process issues. Each maturity level requires a minimum number of yes answers, which must be backed by documented evidence. The questionnaires represent about 10-20 percent of collected information; structured interviews and other methods are used to gather the remaining information.

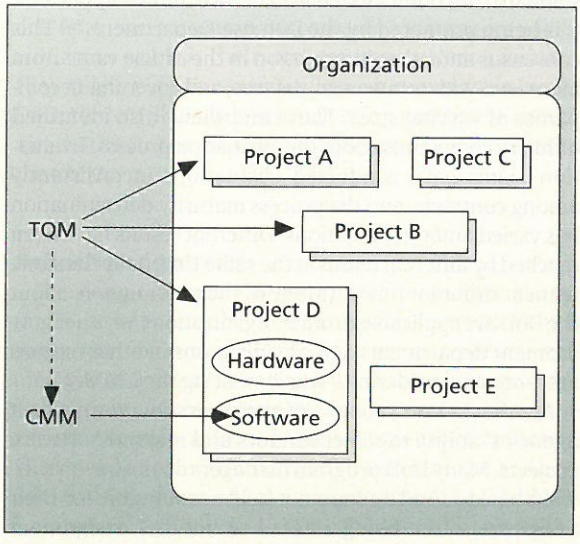



**Figure 2. Relationship between Capability Maturity Model and Total Quality Management.**

Most companies (approximately 80 percent of companies assessed) are currently at Level 1. Consequently, the Level 1 category includes a very broad organizational range, from organizations totally incapable of producing software to those with excellent bottom-line development track records. Some companies are rated at Level 1 because they miss important Level 2 questions, even though they may be able but are not allowed to affirmatively answer many or all of the questions required for a higher level. If an organization misses more than one of the 12 key questions for entry to Level 2, it fails the entire test regardless of how well it can answer the other questions. Therefore, the rating is viewed by some as quite arbitrary.

Humphrey and Curtis[11] responded to several faults identified by Bollinger and McGowan.[4] One interesting statement from this response contradicts planned government contract award procedures: "The fact is that the

Software Engineering Institute instructs Software Capability Evaluation auditors not to base their contract-award recommendations on maturity grades for software vendors." But this article's opening paragraph states, one government organization is intending to mandate that bidding vendors be at Level 3. Whatever the intention, prudent software vendors must assume that certain levels will be mandated or at least will become the de facto standard for bidder selection.

**ONE NEARLY UNIVERSAL COMPLAINT IS THAT MOVING FROM LEVEL TO LEVEL CAN COST HUNDREDS OF THOUSANDS OR EVEN MILLIONS OF DOLLARS.**

This assumption is supported by an article published in *Signal* (the official journal publication for the Air Force Communications and Electronics Association) stating, "The general consensus among those interviewed is that the SEI model eventually would become the process standard over other models in existence, primarily because it is being promoted by the Defense Department."[12] This consensus and other information in the article came from interviews with commercial, defense, and government companies of varying sizes. These interviews also identified industry complaints about the evaluation process. Evaluation teams have conducted evaluations inconsistently among contracts, and the process maturity determination has varied among evaluations. Different results have been reached by different teams at the same time or by the same team at different times. (Many of these comments about the DoD are applicable to other organizations, as other government departments and organizations in other nations are—or are considering—implementing the CMM.)

Another industry complaint questions some government agencies' ability to select vendors and manage software projects. Many DoD program managers do not have a software background and are not held accountable for their programs, often being rotated to another assignment before program completion.[12] This sometimes results in a Level 2, 3, or 5 organization being managed by a "Level 0" government bureaucracy. However, there is some optimism that DoD is becoming more educated about the CMM and that improvements in this area are forthcoming.

Another complaint about the CMM and the resultant SCE is that the evaluation team may come from a company that, itself, is a bidder on government software projects (hopefully on projects unrelated to those projects evaluated). Hence, companies may be reluctant to welcome the evaluation team, fearing that the promise of confidentiality may be subverted.

One nearly universal complaint is that moving from level to level can cost hundreds of thousands or even millions of dollars. Since the government mandates levels and simultaneously selects bidders using lowest cost as a significant criterion, this creates a very real dilemma. In addition, government financial assistance is probably not forthcoming during this time of reduced budgets. Although the ROI may eventually become positive, organizations can incur expenses and reduced profits for years before increased efficiency offsets the cost.

## RECENT INSIGHTS

We discussed the CMM's impact with several companies. However, many companies are reluctant to divulge specific material. To protect the interests of companies from which information *was* obtained, we've omitted company names and other revealing details. Furthermore, since we usually spoke with one knowledgeable manager at each company, our results do not necessarily represent the official company position.

It is difficult to determine whether a company is using the CMM in a serious effort to improve software development or merely to be competitive on upcoming government contracts. The former is driven by product quality, the latter by near-term business considerations. Both motives can be valid simultaneously in a single company.

Beliefs vary as to whether and when a specific CMM level will become a requirement for bidding on government contracts. One opinion is that requiring a specific CMM compliance level is against the principles under which the CMM was developed and, therefore, compliance can only be used as one criterion for bidder selection. But it is probably dangerous to rely on this assumption, since a specific maturity level may have very heavily weighted criteria. Another possibility is that those responsible for bidder selection may rightfully view increasing CMM levels as increasing their ability to mitigate "risk," an important factor in bidder selection since it affects bidder ranking.

Companies also reported that achieving a higher CMM level is a very complex process. However, since most companies are at Level 1, we discuss experiences related to progressing from Level 1 to Level 2. Since process improvement is essentially level independent, insight gained from these experiences should be applicable for promotions to other levels. Nevertheless, we must stress that we uncovered no information on the relative effort needed for each level promotion. There are no known guidelines that indicate, as a function of efforts required to achieve Level 2, the amount of additional effort required to achieve Levels 3, 4, and 5.

### Implementation approaches

Meetings, meetings, and more meetings seem to be the norm for beginning to achieve the first CMM level increase. Since the companies we interviewed were in the formative stages of implementing CMM principles, they expected action and results to replace planning and meetings (hopefully rapidly) . But meetings to assess progress will likely continue, because the companies we queried had not yet experienced the action or implementation stage.

CMM software development principles, like most software engineering principles, stress initial analysis and design. Companies reported that CMM-mandated processes were more likely to succeed when implemented in phases, with significant initial analysis and design efforts.

One company obtaining new major contracts is initiating them with CMM Level 2 principles even when these principles are not mandated for those specific contracts. The reasoning is that future assessments or evaluations will require certain projects to be at the specified CMM level, and establishing new contracts on a solid CMM foundation will improve the company's overall posture. A par-

allel effort involves identifying some key current contracts and making them comply with CMM Level 2.

It may appear that there is high potential for a conflict of interest when a company is assessing itself. But honesty is rewarded, as a self-assessment is often followed by an external, impartial evaluation (an SCE). Moreover, because of the personnel involved (line personnel and first-level management) the focus tends to be on technical and practical issues. Consequently, the CMM goal of improved product quality can indeed offset the temptation to present a facade for near-term contract awards.

**PROCESS IMPROVEMENT WORKS BEST WHEN BOTH EMPLOYEES AND EMPLOYER AGREE TO ACCEPT THE REQUIRED EXTRA EFFORT AND EXPENSE.**

## Benefits and impacts

Published studies of software engineering improvements measured by the CMM indicate significant cost savings or profit return. This implies that software testing and maintenance costs were reduced, since the software better meets verification and validation requirements.

In their analysis of CMM-compliance costs, companies distinguished (1) one-time costs of achieving a higher CMM level from (2) continuing costs of performing software engineering at that higher level.

The latter may actually represent a cost reduction when compared to software production costs at a lower CMM level. Some studies have even shown that the one-time cost of achieving a higher level are quickly recouped by the significant savings of producing software at the higher CMM level.

Companies reported that process improvement works best when both employees and employer agree to accept the required extra effort and expense. One of many possible arrangements is to have some meetings or training conducted during lunch hour, with the employer providing lunch. Other variations and employer/employee compromises include "shared time," when training is done on 50-percent company time and 50-percent employee time.

Much has been said about how an employer or company benefits from implementing the CMM, but little has been stated about how employees benefit. The techniques learned are useful professional skills. The higher CMM level in which the employee works, the more valuable the employee is to the computing industry. This type of expertise can be very marketable. In addition, employee pride and management respect should not be overlooked as an employee benefit, reward, or motivating force.

The companies we questioned agreed that reputation with their customers is primarily based on product quality and agreeable interface with those customers. There is little argument that higher CMM levels should lead to better quality software and therefore better company reputation. However, CMM compliance may also change the manner in which a company interacts with its customers. For example, the formalism of higher CMM levels will make ad hoc contractor responses to volatile customer demands more difficult, but will contribute to more reliable and mutually beneficial contractor-customer relationships. Fortunately, the most compelling argument is also a very simple one: Higher quality software at lower cost along with improved company reputation should be a very potent formula for winning and keeping contracts.

SINCE NO APPROACH THAT ENFORCES IMPROVEMENTS will be universally acceptable in all aspects to all concerned, the CMM, on balance, can be considered a very successful model, particularly when combined with TQM principles. What may be less certain is whether the costs of attaining and maintaining a CMM level will be recouped through reduced software production costs and more efficient software engineering practices. Published studies (some cited in this article) report that process improvement based on the CMM more than pays for itself. However, we can safely assume that far more studies will report a positive outcome than a negative outcome. Few companies will be willing to publish their process-improvement failures. Nevertheless, with continued strong sponsorship by the DoD, there will likely be an increasing number of companies that base their software process improvement efforts on the CMM. ■


## Acknowledgments

The comments and suggestions of *Computer*'s anonymous referees have been very helpful. Corrections and suggestions by Richard Bechtold, a senior technical staff member at Software Productivity Consortium, substantially improved this article's overall content and organization. We also acknowledge comments from Laurie Werth, who reviewed the revised version; she has produced an excellent educational module (SEI-93-EM-8) from which Table 2 came. Mark Paulk kindly offered his time to review the article with one of the authors and answer many questions.


## References


1. W.S. Humphrey, "Introduction to Software Process Improvement," Tech. Report CMU/SEI-92-TR-7, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, June 1993.
2. H. Krasner et al., "Lessons Learned from a Software Process Modeling System," *Comm. ACM*, Vol. 35, No. 9, Sept. 1992, pp. 91-111.
3. M. Paulk et al., "Capability Maturity Model for Software, Version 1.1," Tech. Report CMU/SEI-93-TR-24, Feb. 1993.
4. T.B. Bollinger and C. McGowan, "A Critical Look at Software Capability Evaluations," *IEEE Software*, Vol. 8, No. 4, July 1991, pp. 25-41.
5. W.S. Humphrey, T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software*, Vol. 8, No. 4, July 1991, pp. 11-23.
6. R. Dion, "Elements of a Process Improvement Program," *IEEE Software*, Vol. 9, No. 4, July 1992, pp. 83-85.
7. W.H. Lipke and K.L. Butler, "Software Process Improvement: A Success Story," *CrossTalk*, Nov. 1992, pp. 29-39.
8. D. Card, "Capability Evaluations Rated Highly Variable," *IEEE Software*, Vol. 9, No. 5, Sept. 1992, pp. 105-106.

9. T. Pyzdek, "To Improve Your Process: Keep it Simple," *IEEE Software*, Vol. 9, No. 5, Sept. 1992, pp. 112-113.
10. B. Silver, "TQM vs. the SEI Capability Maturity Model," *Software Quality World*, Vol. 4, No. 2, Dec. 1992.
11. W.S. Humphrey and B. Curtis, "Comments on 'A Critical Look,'" *IEEE Software*, Vol. 8, No. 4, July 1991, pp. 42-46.
12. D.L. Johnson and J. Brodman, "Software Process Rigors Yield Stress, Efficiency," *Signal*, Vol. 46, No. 12, Aug. 1992, pp. 55-57.

***Hossein Saiedian*** *is an assistant professor in the Department of Computer Science at the University of Nebraska at Omaha. His research interests include software engineering models, formal methods, and object-oriented computing. He received his BS in information systems in 1981 and MS in mathematics in 1984 from Emporia State University, and his PhD in 1989 in computing and information sciences from Kansas State University. He is a member of the IEEE Computer Society, Sigma Xi, and the ACM, and is chair of the ACM Special Interest Group on Individual Computing Environments (SIGICE). He was recently ranked as the third leading software engineering scholar by the* Journal of Systems and Software *(October 1994).*

***Richard Kuzara*** *manages a project related to the AutoDIN (Automatic Digital Information Network) Communications Support Processor system designed and maintained by Sterling Software in Bellevue, Nebraska. For the last 32 years, he has been directly involved in the development of computer systems, and most of his experience has been with US military intelligence systems. He received his BA in mathematics in 1963 from the University of Wyoming and his MS in computer science in 1994 from the University of Nebraska at Omaha. He is a member of the Armed Forces Communications and Electronics Association (AFCEA).*

*Readers can contact the authors at the Department of Computer Science, University of Nebraska at Omaha, Omaha, Nebraska 68182; e-mail hossein@unocss.unomaha.edu.*