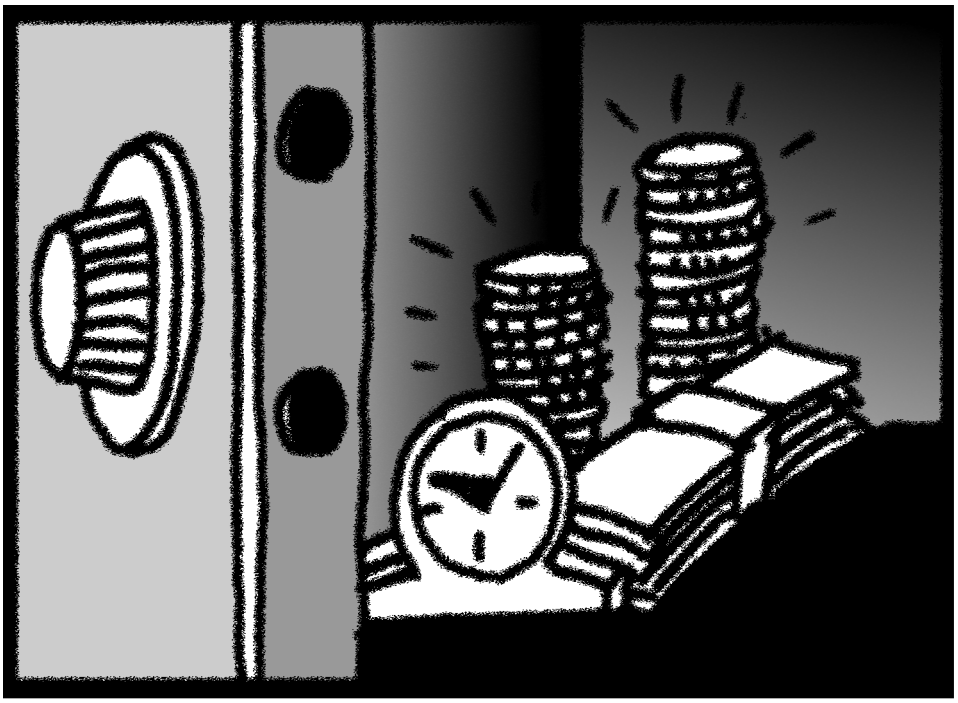
How Software Process Improvement Helped Motorola

MICHAEL DIAZ AND JOSEPH SLIGO
Motorola SSTG



Many organizations are using or considering the Capability Maturity Model as a vehicle for software process improvement. But does the CMM provide real benefits? The authors offer metrics and data that show the results of Motorola's CMM usage.

In many companies, the Capability Maturity Model[1] plays a major role in defining software process improvement. Frequently, organizations contemplating software process improvement (SPI) seek assurances that tangible benefits will result from such activities. Pockets of data across industry[2] show that CMM-based process improvement is making a difference in those organizations committed to improvement. Raytheon yielded a twofold increase in its productivity and a return ratio of 7.7 to 1 on its improvement expenditures, for a 1990 savings of $4.48 million from a $0.58 million investment. Over a period of four and a half years, from mid-1988 to the end of 1992, the company eliminated $15.8 million in rework costs.

0740-7459/97/$10.00 © 1997 IEEE

TABLE 1
MOTOROLA GED PROJECT PERFORMANCE BY SEI CMM LEVEL.

SEI CMM Level	Number of Projects	Quality (In-Process Defects/ MAELOC*)	Cycle Time (X factor)	Productivity (Relative)
1	3	n/a	1.0	n/a
2	9	890	3.2	1.0
3	5	411	2.7	0.8
4	8	205	5.0	2.3
5	9	126	7.8	2.8

*Million assembly-equivalent lines of code

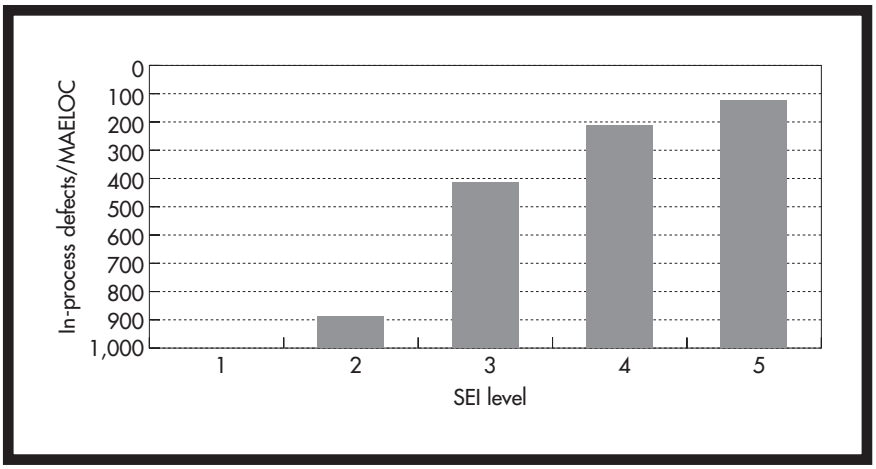


Figure 1. *Quality by SEI level, with quality defined by the scarcity of in-process defects.*

Hughes Aircraft has computed a 5-to-1 return ratio for its process improvement initiatives, based on changes in its cost–performance index. This has generated an annual savings of about $2 million above Hughes' process improvement expenditures. Tinker Air Force Base recently computed a 5-to-1 return on investment for its process improvement initiatives, which generated a savings of $3.8 million from a $0.64 million investment. Other sources confirm this trend.[3,4]

Motorola has long been a champion of the SEI's CMM as a vehicle for furthering software process improvement. In November 1995, the company's Government Electronics Division was independently assessed at SEI level 4. The assessment further rated GED's policies and procedures at SEI level 5, without verifying project implementation of the level 5 key process areas. Motorola GED employs about 1,500 engineers to design and build a wide variety of government electronic systems. Approximately 350 GED engineers participate directly in software development. Our organization's long history of support for process improvement was, we found, to prove key in reaping benefits from the CMM process. In effect, we supplemented the SEI's model with several of our own programs.

The cycle time, quality, and productivity metrics we use are based on current measures from approximately 34 programs, which are at various stages in the software life cycle. Motorola GED plans to reevaluate this project performance analysis when all projects are completed.

PROCESS METRICS

At Motorola GED, each project performs a quarterly SEI self-assessment. The project evaluates each key process area (KPA) activity as a score between 1 and 10, which is then rolled into an average score for each KPA. Any KPA average score that falls below 7 is considered a weakness. The SEI level for the project is defined as the level at which all associated KPAs are considered strengths, that is, all KPA average scores must equal 7 or more.

GED uses quality, cycle time, and productivity to evaluate development programs because our customers value these attributes. In addition, Motorola has always valued quality in all its products and processes: its Six Sigma Quality focus has been a corporate initiative for several years. Six Sigma is a quality process that looks at reject rates as low as a few per million opportunities. This process originally started in the manufacturing arena and has been expanded to software at Motorola.

Recently, Motorola corporate has been championing the 10X cycle-time initiative, which seeks to have all business elements achieve a 10-fold reduction in product cycle time to accelerate the introduction of new products. Productivity is directly related to our ability to win new programs from our traditional US Department of Defense customer and drives our profitability in emerging commercial products. Table 1 summarizes the Motorola GED improvement trends for quality, cycle time, and productivity by SEI level. Motorola obtained performance data in these areas for each program from its internal metrics, and categorized them by SEI level as determined by each project's internal self assessment. The projects involved in the analysis are at various stages of development.

A more detailed examination of each

metric follows, beginning with the empirical methodology by which the metric was derived. Specific improvement results are not entirely attributable to increasing SEI maturity levels, because Motorola GED has put into place significant cycle time and quality initiatives above and beyond the SEI CMM.

Quality. A problem detected in the same phase it was introduced is defined as an *error*; a *defect* is a problem that escapes detection in the phase it was introduced. The *quality* metric is defined as defects per million earned assembly-equivalent lines of code (defects/MEAELOC). AELOC, the *assembly-equivalent lines of code*, equals delivered source instructions × the Capers Jones Expansion Factor.[5] The *earned* AELOC, or EAELOC, is defined as AELOC × percent complete, which equals the budgeted cost of work performed/budget at complete.

Results. Figure 1 examines the quality of each project categorized by the project's internal SEI self-assessment. Since most of these programs are still active, we derived the normalization factor by using the EAELOC.

Analysis. Project results show that each level of the CMM improves quality by a factor of about 2. The improvement in quality is expected for projects that transition from level 2 to 3 due to the Peer Review KPA found in level 3. Peer reviews have been widely recognized in the industry for being the single most important factor in detecting and preventing defects in software products. Quality is also expected to improve for projects transitioning from level 3 to 4 due to the Quantitative Process Management and Software Quality Management KPAs. Using quality metric data such as phase containment effectiveness (the ratio of problems inserted and detected within a phase to the summation of all problems introduced in that phase) will let projects modify their processes when the observed metric falls below the organizational control limits.

For example, if the peer review process detects 75 of every 100 problems introduced during detailed design, the phase containment effectiveness would be 75 percent. You can estimate the number of problems introduced by using historical defect density data from similar projects and tracking problems found early in the development cycle. (A method to predict problems throughout the development cycle is given elsewhere.[6]) The Motorola GED control limit is at 85 percent phase containment effectiveness. Projects below this threshold perform causal analysis to improve their peer review and testing processes. We attribute the improvement from level 4 to level 5 to the Defect Prevention and Process Change Management KPAs. Projects operating at this level perform Pareto analysis on the root cause of their problems and perform causal analysis to determine the process changes needed to prevent similar problems from occurring in the future.

You can more readily achieve large improvements in defect density when the number of defects is large, as would be expected in lower–maturity-level projects. At higher maturity levels, it becomes increasingly difficult to dramatically reduce the defect density.

Cycle time. We define the X factor or cycle-time metric as the amount of calendar time for the baseline project to develop a product divided by the cycle time for the new project. For example, if the baseline project took six months to complete, and the new project took two months to complete, then the new project's X factor would be 3. Motorola's goal is to achieve an X factor of 10 for new projects within five years.

The project we used as our baseline was a similar project completed at Motorola GED prior to 1992. Each project at GED tracks its cycle time by selecting a program with a similar target domain and by tracking its progress against that baseline program. The cycle-time ratio of the current program over the baseline program is known as the program's X factor. Many times the completion of a program is not entirely under the contractor's control, especially with government funding curve slowdowns. In addition, government contracts typically involve a qualification test cycle that is not normally seen in the commercial world. To more accurately reflect the potential cycle time from a commercial viewpoint, cycle time is measured with respect to a project's first increment. This should eliminate the effect of qualification testing and government funding changes on a project and more accurately compare Motorola GED with the rest of the commercially oriented company.

Results. Figure 2 shows the cycle-time improvements with respect to SEI level. Because the X factor is derived by dividing an older, baseline project's completion time by the current project's com-

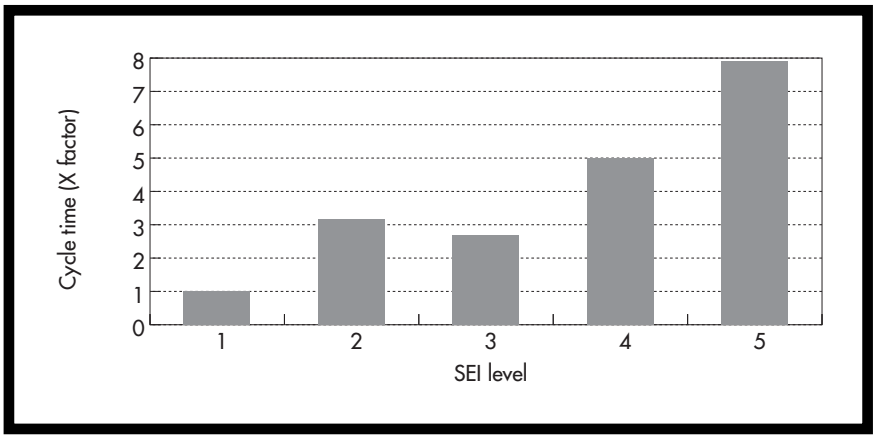


Figure 2. *Cycle time by SEI level. Shorter cycle times result in larger X factors.*

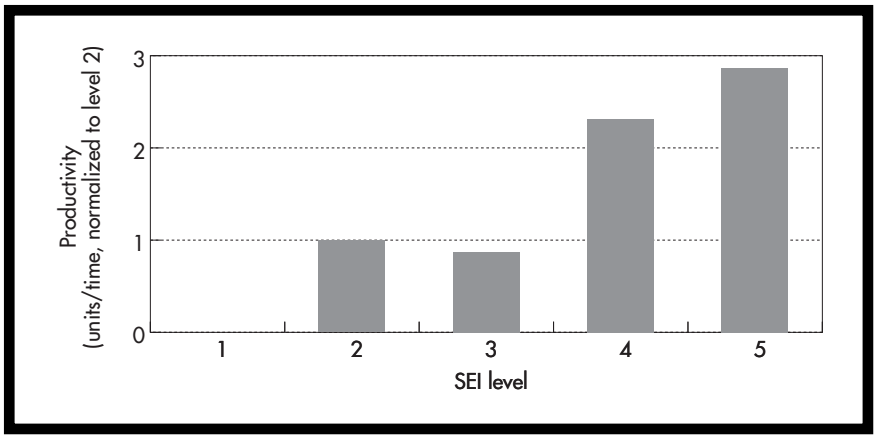


Figure 3. *Relative productivity by SEI level, normalized to the productivity of an average level 2 project. We define productivity as the number of assembly-equivalent lines of code produced, divided by the time required to produce them.*

pletion time, the shorter the current project the larger the X factor.

Analysis. Analysis of the data shows a 3.2 X factor for projects transitioning from level 1 to 2, a surprising decrease in cycle time for projects that move from level 2 to 3, an increase from 2.7X to 5.0X for projects that move from level 3 to 4, and an increase from 5.0X to 7.8X for projects that move from level 4 to 5. The decrease in cycle-time X factor for projects moving from level 2 to 3 may indicate a weak correlation between schedule performance and maturity level that has been seen in other CMM benefit surveys. Elsewhere, we have noted a weak schedule performance index correlation to maturity levels above level 1.[4] At least one other survey, however, does not evince such a correlation.[3]

The upward trend in cycle time above level 3 tends to corroborate the underlying assumption that higher-maturity projects have a better schedule performance index.

Motorola's 10X initiative is separate and in addition to the SEI CMM level 5 initiative. A major component of the 10X initiative is the implementation of a life cycle called incremental development. The idea is to complete a thread of functionality that will test all the system interfaces and demonstrate some functionality that has meaning to the customer. Because the effort to progress from level 2 to level 3 involves a significant number of KPAs and process changes, it is expected that such a project may also have trouble absorbing a new life cycle such as incremental development.

Because many of the projects used as a basis for this analysis are still in development, further analysis with final schedule performance data is required to corroborate the preliminary findings. Another attribute that seems to correlate with increasing SEI maturity level is the decrease in variability of schedule and cost performance. The performance to plan variability decreases with higher maturity levels.

Productivity. We define *productivity* as the amount of work produced divided by the time to produce that work. This may be measured in source lines of code per hour or a similar measure. Each project at Motorola GED tracks its productivity by measuring AELOC produced and the number of hours needed to produce that code.

Results. For proprietary reasons, we do not show the actual number of lines of code per hour. Figure 3 does, however, show the relative productivity between projects at different levels of maturity. The data is normalized to the productivity of an average level 2 project.

Analysis. Factors other than process maturity affect productivity, most importantly technology changes. For example, the data shown includes projects that may have started before some form of automated code generation became available. In addition, the amount of code reuse on a project can greatly affect that project's productivity. As projects increase their level of maturity, the organization is better able to effectively reuse software source code. Likewise, software code that is reused from a high-maturity-level project requires less rework and is more easily understood. These factors act as multipliers in the productivity of high-maturity-level projects.

Projects experience an unexpected decrease in productivity when moving from level 2 to level 3. This appears to be a side effect of asking project staff to do too many new things all at once at level 3. When instituting a level 3 system, new processes are rolled out that greatly affect the way individual project members perform their tasks. This "new process rollout" does not have as great an impact at higher maturity levels. As with any adoption of new technology, we expect that an absorption cycle will be needed before the full benefits can be observed. At levels 4 and 5, each project quantitatively measures its own performance and can effectively change its processes while maintaining productivity.

Yet simply measuring performance does not ensure that productivity can be maintained. First, when you analyze and select any new process improvement, you should do so with the overall goal of at least maintaining productivity if not increasing it. Second, you must then monitor the actual productivity impact of the process improvement you implement. Real-time adjustments to the process improvement may be needed to keep the project working efficiently.

IMPLEMENTATION STRATEGIES

SPI implementation at Motorola GED began in 1989, when GED was assessed at level 2 of the Process Maturity

Model. At this time, we had already met most of the level 2 KPAs by doing what our government contracts required. Yet, while each project performed certain level 2 activities as required, there was no real organizational focus on software engineering processes.

While at level 2, GED's chief software engineers began focusing on process improvement. A high-level standard policy and procedure further defined GED processes for software engineering, as did the publication of the *Software Quality Management Manual*. This manual laid out steps to follow in a typical "waterfall" software development, where requirements analysis led to design, which in turn led to code and testing. At this time a software functional team was created to attempt to unify software engineers from various parts of the organization into a cohesive team. Peer review concepts were also introduced. These activities led to a level 3 assessment of Motorola GED in 1992.

Working groups. Starting in 1992 and continuing through 1994, the engineering organization created a process improvement working group, made up of senior practitioners. In addition, an initial software engineering improvement working group was formed with eight senior task leaders responsible for software development. This group had hands-on leadership from the engineering department manager, which proved critical to its success. It created the burden code metrics tool, which collected the amount and type of effort expended on each project. The group also defined and applied process and quality metrics that were useful for each project. At the end of this period, GED staff created the *Handbook for Quantitative Management of Software Process and Quality*, based on this group's work. These activities resulted in GED being assessed at level 4 of the Process Maturity Model, based on the PMM, with goals added from the CMM.

Since 1994, GED has maintained a defect prevention working group that looks at quality data from around the organization to identify systemic causes of poor quality. This group also created the *Defect Prevention Handbook*, which projects can use to perform their own defect prevention activities. The chief software engineer group was expanded to ensure that all new projects could begin operating at level 5; each project performs a self-assessment of CMM activities to identify areas that need work. During this period, the level 5 metrics tool was created and released to help projects integrate metrics collection from various sources in their level 4 and level 5 activities. In addition, GED expanded the working group from a single team of eight members to approximately four teams, each with a dozen software leaders. The chief software engineers also created handbooks for process change management and technology change management.

Throughout these activities, senior management sponsorship proved critical to the success of the process improvement efforts. This meant not only taking an active interest in the progress of various process improvement initiatives, but also providing funding and time to do the work, and rewarding those who contributed. Ongoing activities include

- continuing review and improvement of existing handbooks;
- maintenance of the process improvement request system;
- weekly meetings of the working groups to address process, technology, and people issues;
- meetings of chief software engineers and all new-project staffs to ensure that level 5 principles are followed; and
- ongoing integration of development and management tools.

Optimizing implementation. From the data gleaned during these activities, we found that several strategies provide optimal results.

- Focus on improving new projects. It is extremely difficult to change projects, especially at a low maturity level, once they have started.
- Adopt a top-down focus before immersing yourself in CMM details; start by assessing the intent of each KPA so that you can determine how it fits into your environment.
- Emphasize productivity, quality, and cycle time. Avoid process for its own sake.
- Management commitment is needed from all levels; commitment from upper management won't be enough unless individual project leaders and managers are also determined to succeed.
- Practitioners and task leaders, not outside process experts, should be used to define processes.
- Managers must be convinced of process improvement's value; it's not free, but in the long run it more than pays for itself.
- The customer must be kept informed about the process, especially when process changes occur.
- Copying process documents from other organizations usually does not work well; the process must match your organization.
- Overcoming resistance to change is probably the most difficult rung to climb on the SEI CMM ladder.

Process change takes time, talent, and a commitment that many organizations are uncomfortable with. Based on our experience, we believe the investment is worth it.

Practitioners and task leaders, not outside process experts, should define process.

ECONOMIC BENEFITS

To evaluate how much we saved, we must first calculate how much we expended on SPI efforts, then evaluate how much the errors and defects caught or eliminated by these processes would have cost us.

The process improvement effort consisted of the following to support the base

of 350 software developers:

♦ four full-time chief software engineers = 48 staff-months;

♦ task leaders of the software engineering improvement working group = 34 projects × 1 hour/week = 10.5 staff-months;

♦ prephase kickoffs and postmortems: 34 projects × 1 hour per phase × 7 phases = 1.5 staff-months;

♦ software development planning: 34 projects × 5 days = 1 staff-month; and

♦ defect prevention working group: 8 members × 1 hour per week = 2.5 staff-months.

The total of all process improvement activities was approximately a 1.5 percent investment of our base staffing.

From a quality perspective, the defect injection rate decreases by roughly half each time a project advances a CMM level. Therefore, an SEI level 2 project has a defect injection rate eight times greater than an SEI level 5 program. The cost of rework is therefore at least eight times greater at level 2. Assuming that each defect requires an average of 16 hours' rework and analysis, and that the cost of rework is approximately $100 an hour, a typical 500,000-AELOC (about 100,000 SLOC) project can anticipate the following savings:

♦ At level 5, 63 defects (based upon 126 defects per MAELOC) would expend $100,800 on rework.

♦ At level 2, 445 defects (based upon 890 defects per MAELOC) would expend $712,000 on rework.

A typical 100,000-SLOC project would span 18 months and employ approximately 20 software engineers. If we allocate the 1.5-percent process improvement cost into each specific project, this would amount to 1.5 percent × 20 people × 18 months × 167 hours × $100/hr = $90,180 investment.

The resulting return on investment would be ($712,000 – $100,800) = $611,200 for a $90,180 investment, or a total return of 677 percent.

Unfortunately, given the government nature of our contracts, Motorola GED does not realize all these savings as profits. The true cost–benefit occurs when projects finish earlier, allowing us to apply more engineering resources to the acquisition and development of new business.

Each level of SEI CMM maturity reduces defect density by a factor of 2. Cycle time and, to a lesser extent, productivity improve with each maturity level except level 3, when cycle time and productivity both decrease. From this, we can conclude that achieving level 3 involves significant new process introduction, which can negatively affect these two metrics until the project absorbs and learns to tailor the processes. From the data presented here, it would appear that setting a goal of SEI level 3 for an organization is the least beneficial point to shoot for.

The effort to transition from level 2 to level 3 is probably the most difficult because of the many KPAs associated with level 3 and the impact that process maturity plays in SPI. Lower-maturity organizations find it much more difficult to change and implement SPI for a variety of reasons.

♦ Keying process changes to metric analysis data is not addressed until CMM levels 4 and 5. Such data is critical to improving the effectiveness of SPI efforts.

♦ Lower-maturity organizations focus on defining their core processes, not on improvement.

♦ Lower-maturity organizations are just starting to improve their software processes. This requires significant effort, especially in the beginning. Staff skepticism can also be an obstacle. Before they buy into a new SPI initiative, most software engineers will wait to see if it truly has management support and staying power.

These factors suggest that the SEI CMM could be improved by addressing some aspects of all KPAs even at the lower maturity levels. For example, some aspects of defect prevention and process improvement can be performed at the lower maturity levels. The ISO Spice model is an example of such a graduated approach.

Process improvements take time to institutionalize and require a commitment from management to succeed. Achieving higher levels of process maturity requires an investment of time and money in process improvements, including tool integration to aid in the collection and interpretation of quantitative data.

Given the costs they incur, process improvement activities must be undertaken with a view to return on investment. You could easily set up a high–SEI-maturity organization that would suffer slowed delivery times and reduced productivity if process were followed for process's sake. Thus, in addition to the traditional SEI CMM emphasis, we must tailor our processes and focus on cycle time. ◆

REFERENCES

1. M. Paulk et al., "Capability Maturity Model Version 1.1," *IEEE Software*, July 1993, pp. 18-27.
2. J.G. Brodman and D. Johnson, "Return on Investment from Software Process Improvement as Measured by U.S. Industry," *Crosstalk*, Apr. 1996, pp. 23-28.
3. J.D. Herbsleb and D.R. Goldenson, "A Systematic Survey of CMM Experience and Results," *Proc. ICSE 18*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1996, pp. 323-330.
4. P. Lawlis, R. Flowe, and J. Thordahl, "A Correlational Study of the CMM and Software Development Performance," *Crosstalk*, Sept. 1995, pp. 21-25.
5. C. Jones, *Applied Software Measurement*, McGraw Hill, New York, 1991, 1996.
6. F. Arkell and J. Sligo, "Software Problem Prediction and Quality Management," *Proc. 7th Int'l Conf. Applications of Software Measurement*, Software Quality Engineering, Jacksonville, Fla., Oct. 1996.

Michael Diaz is the manager of Security/Integrity and Quality for the Motorola Information Security Division. His areas of focus are software process improvement, metrics, security countermeasures, and risk man-agement. Diaz was the chief software engineer for Motorola GED from 1994 to 1996. Diaz's previous experience includes 15 years of software technical leadership in requirements management, systems engineering, security architectures, and secure key management systems at Motorola. Diaz has been awarded membership in Motorola's Scientific Advisory Board Association, the highest technical association within Motorola.

Diaz received a BSEE and an MS in computer engineering from Boston University. He is a member of the IEEE Computer Society and the Society for Hispanic Engineers (SHPE).

Joseph Sligo is the chief software engineer for Motorola ASD, responsible for all aspects of software development in an organization of 350 software engineers. His areas of focus include software process improvement and project management. Sligo's previous experience includes 15 years of software technical leadership for GPS receivers, spaceborne advanced digital communications transponders, and position determining systems. Sligo has been awarded membership in Motorola's Scientific Advisory Board Association, the highest technical association within Motorola. His experience includes work on Omega navigation receivers and instrument landing systems with the Ohio University Avionics Research Center.

Sligo received a BSEE from Ohio University.

Address questions about this article to Diaz at Motorola GED, 8201 E. McDowell Rd., Scottsdale, AZ 85252; p17114@email.mot.com; or to Sligo at the same address; p16628@email.mot.com.

CALL FOR CONTRIBUTIONS

Ninth IEEE International Workshop on Software Specification and Design (IWSSD-9)
Ise-shima, Japan
April 16-18, 1998

In conjunction with International Conference on Software Engineering 1998

The International Workshop on Software Specification and Design (IWSSD) is a leading international forum for research on software architecture, concurrent, distributed and real-time systems, formal models, and requirements and design methods. As with previous meetings, IWSSD-9 will use working-group discussions in order for participants to focus on their shared concerns in representing and reasoning about models of software-intensive systems. Workshop attendance is by invitation only, based on a submitted paper.

The program committee is requesting the submission of research papers outlining novel research directions, research projects or research proposals. The papers should make a case for the significance and originality of the work and clearly describe (preliminary) results, infra-structure or exploratory studies on which the work is based. Of particular interest is the rationale underlying the work particularly if it identifies gaps in research and difficulties that have not been well understood. Papers should not exceed eight pages, including figures.

Submit six copies for receipt by November 14th 1997 to:

Prof. Ugo Buy
EECS Dept. (M/C 154)
University of Illinois
851 South Morgan Street
Chicago, IL 60607
USA
EMAIL: buy@figaro.eecs.uic.edu

Additional submission information, including information about electronic submissions, can be found at URL: http://salab-www.cs.titech.ac.jp/iwssd9.html.