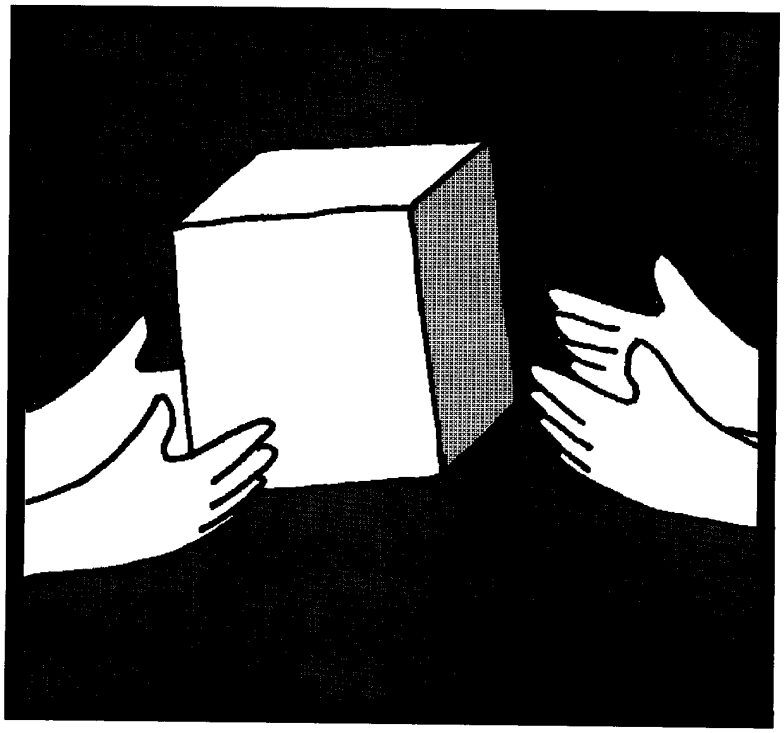
FEATURE

TECHNOLOGY TRANSFER AT MOTOROLA

While developing a formal software-review process, a working group at Motorola devised a technology-transfer model that is built on process packages, each one targeted to a different user group. Their model allows for tailoring, makes training and consulting widely available, and relies on champions.

VICTOR R. BASILI
University of Maryland
MICHAEL K. DASKALANTONAKIS and
ROBERT H. YACOBELLIS, Motorola



Although new processes, methods, and tools are introduced in the literature every year, few are actually adopted. Development managers in industry complain that these new ideas are either not applicable to real-world projects or that their process is not mature enough to incorporate them.

Consider a project manager who buys a tool to improve change control. The tool is virtually worthless without a well-defined, documented, and reasonable change-control process, and even if there is such a process the development team is likely to need training in both the process and the tool before they can be used on a real project. But too often the manager fails to allocate sufficient training time and doesn't anticipate the initial drop in productivity. This situation occurs time and time again.

We believe part of the problem is that the industry lacks a focused, needs-based approach to tailoring and transferring software-engineering technology. At Motorola, we have developed an approach that helps development organizations focus on the technology they really need, devise solutions, and transfer those solutions to development teams. In this article, we report our experience using this approach in the last five years and the lessons we learned.

TARGETED PROCESS PACKAGES

Through 15 years of study at the US National Aeronautics and Space

0740-7459/94/$04.00 © 1994 IEEE

Administration's Software Engineering Laboratory and elsewhere, a set of software-engineering technology principles has evolved[1] that recommend organizations

- Develop quality-focused software-engineering technology within a business unit.
- Formalize plans to tailor, transfer, and deploy software-engineering technology.
- Evaluate software-engineering technology to improve it on the basis of feedback obtained from goal-oriented measurement.

Experience in applying these principles, in turn, has produced recommendations for measuring software processes and products in the context of software-engineering technology:

- Conduct goal-oriented, top-down measurement of processes and products.
- View the measurements and their interpretation from an appropriate perspective.
- Account for differences in project environments, processes, products, and available technology.

These principles and recommendations are embodied in two paradigms: the Quality Improvement Paradigm,[2] a three-part process-improvement approach, and the Goal-Question-Metric paradigm,[2] a mechanism that the Quality Improvement Paradigm incorporates for establishing project and corporate goals and measuring against those goals.

Motorola's corporate-wide Metrics Working Group adopted the Quality Improvement Paradigm and instantiated it with a set of organizational procedures to identify, tailor, and transfer software-engineering technology. The Motorola version is called the Software Engineering Improvement Paradigm. It is designed to help managers focus on software-engineering technology as it applies to specific development activities, such as testing, product reviews, and management. It also provides a justification for selecting and tailoring software-engineering technology to individual projects and a mechanism for evaluating technology against a business unit's goals.

Fundamental to our approach is the *process package*, a set of documents and training material that communicates everything about the technology you are trying to transfer. A process package includes an overview of what to expect, how to use the information, references to other corporate efforts and process packages, guidelines for using the process, training aids targeted to different user groups, a set of slides for conducting training workshops, and data and lessons learned.

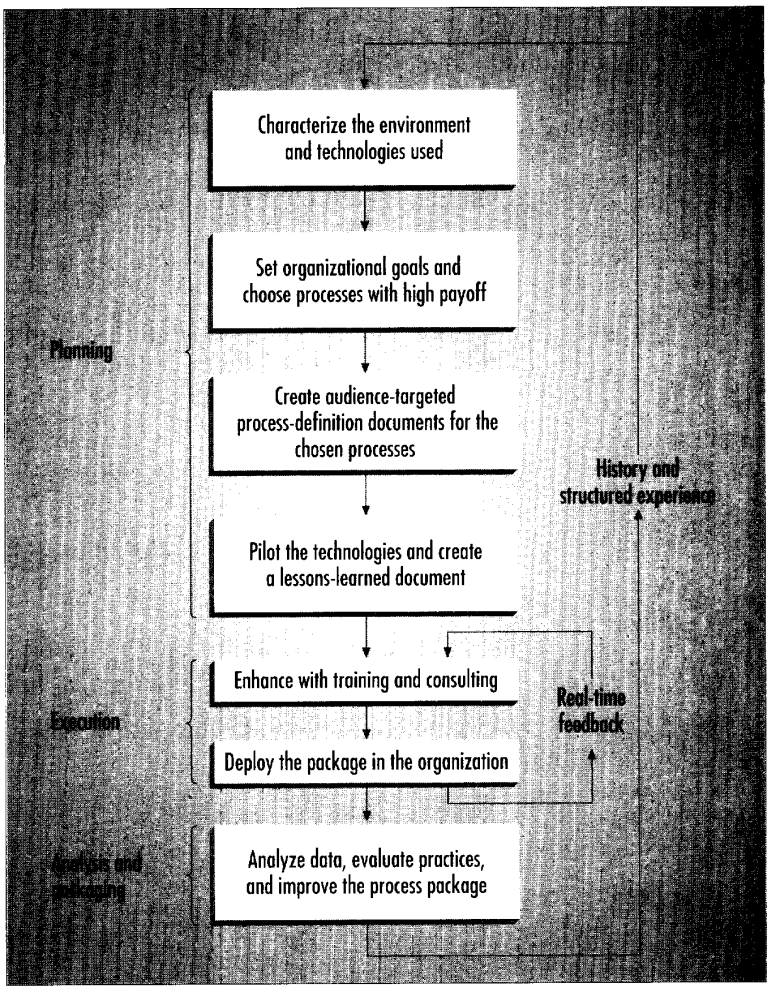


Figure 1. *Evolution of a process package. We built on the Quality Improvement Paradigm's planning, execution, and analysis and packaging phases, then defined seven steps to transferring technology via process packages.*

As Figure 1 shows, a process package evolves over time as experience is gained and feedback is incorporated. Our approach builds on the Quality Improvement Paradigm's three phases: planning, execution, and analysis and packaging. Within these three phases, we defined seven steps:

1. Characterize and evaluate the organization's current environment and technology.
2. Set organizational goals and refine them into quantifiable questions and metrics. Choose the processes that have the best chance of paying off if technology improvements are made.
3. Create documents, targeted to different audiences, that define new technology or improvements to existing technology in those high-payoff areas.
4. Pilot the technology in sample projects, analyze the data, refine the technology, and create a lessons-learned document.
5. Enhance the process package by targeting the training materials and consulting support to a particular audience.
6. Deploy the technology within a business unit, monitor its use carefully, and learn

MOTOROLA'S THREE-STAGE FORMAL SOFTWARE-REVIEW PROCESS

Many software projects in industry use reviews to detect problems early. However, the degree to which they are an integral part of the development process and their effectiveness varies widely. At Motorola, we felt there was a need to formalize the review process and its measurement to maximize its effectiveness and efficiency.

Three stages. The review process package defines a three-stage formal process:

1. Reviewer preparation. Participants agree that the material is ready, select a leader and a review team, schedule a meeting, prepare material, have an optional orientation meeting, study the material and inform the review leader of faults found, and decide if they should hold a meeting or if additional preparation is needed.

2. Review meeting. The review leader introduces the reviewers and their roles and outlines the purpose of the review. Then the presenter starts and reviewers ask questions to expose problems, the author of the reviewed material answers with clarifications only, the recorder takes notes, and the presenter starts again in a loop until the review disposition is determined. The recorder completes a report documenting the review disposition and faults found.

3. Follow-up. The review meeting report is published, the developers fix errors and defects, the recorder fills out a software process-assurance form that summarizes metrics data so that it can be used to improve the process, and the review leader ensures follow-up and schedules any additional reviews.

We based this process description on existing practices within several groups in Motorola and published work.[1-2] However, we tailored our process to address some major issues we identified, such as reviewer preparation.

The package also contains a set of guidelines aimed at enhancing the effectiveness and efficiency of reviews. We developed and continuously improved them by analyzing the data collected for the review metrics defined.

Review goals. An integral part of the review process is the collection and analysis of metrics data for improving not only the review process, but also the development process and the product being reviewed.

We used the Goal-Question-Metric approach to establish quantitative goals for the review process, define the measurements that must be taken to evaluate its effectiveness, discover problems, and improve it. Although this use of GQM was tailored to our priorities, our experience is applicable to projects with different priorities.

Our primary measurement goal was

- Analyze the review process to improve its effectiveness in removing faults, from the corporation's point of view.

Our secondary goals, which used the same data, were

- Analyze the constructive and analytic process of the previous development phases to improve their ability to generate a fault-free product, from the corporation's point of view.
- Analyze the construction process of the current development phase to improve its ability to generate a fault-free product, from the corporation's point of view.
- Analyze the product before and after the review to evaluate its correctness, from the project manager's point of view.

GQM requires that you characterize the environment of the specific project to provide a framework for comparison and to expose other factors that may influence behavior. Sample factors include the number of software engineers on the project; their average expertise and familiarity with the application domain; the application domain and its difficulty; development techniques, tools, and hardware; estimated project size; and target machine.

Review metrics. Using GQM to define metrics involves mapping the measurement goals to sets of questions, which in turn generates supporting questions and defines the metrics that should be collected during a review.

Some of the metrics we defined to evaluate the review process are

- *Review Process Compliance.*

TABLE A
ANALYZING METRICS TO DETERMINE PROJECT QUALITY AT END OF PHASE

Conditions				Facts				
RPC	RPD	ET	DT	Score	Product-in	Product-out	Process-previous	Process-current
C	S	LS	LS	5	good	good	effective	effective
C	S	H	LS	4	good	fixed-up	effective	not-effective
C	S	LS	H	2	poor	fixed-up	not-effective	effective
C	S	H	H	1	poor	fixed-up	not-effective	not-effective
N	x	x	x	0	?	?	?	?
x	U	x	x	0	?	?	?	?

x = don't care value
? = no value due to insufficient information

product-in = product received after review and changes in previous phase
product-out = product generated by current phase after review and changes
process-previous = construction and review process used in previous phase
process-current = construction process used in the current phase

A subjective determination of how well the constructive process and subsequent review has complied with the review process. This subjective determination is done by the person having the SQA perspective. An RPC value is either C (compliant) or N (noncompliant).

♦ *Review Process Domain.* A subjective determination of how well the reviewers understood the document (based on their level of experience and the perspective they represent). This subjective determination is done by the person having the SQA perspective. An RPD value is either S (satisfactory) or U (unsatisfactory).

We defined several metrics for evaluating faults in a product as well as the effectiveness of the fault-removal activities (an error is a fault found during a formal review of a deliverable; a defect is a fault found after the formal review of a deliverable).

♦ *Phase-Containment Effectiveness* is an objective determination of the effectiveness of the reviews of the deliverables produced during a specific project phase. It is defined as the ratio of errors found in reviews to the sum of errors found in reviews and defects that escaped such reviews. The value of PCE is expressed as a percentage, where 100 percent is the best.

♦ *Error Trend* is an indicator of how the normalized number of errors found in a review compares with the corresponding number for past similar projects (those with similar environmental characteristics). The value of ET is either H (higher) or LS (lower, about the same).

♦ *Defect Trend* is an indicator of how the normalized number of defects found during a review in deliverables from previous phases compares with the corresponding number for past similar projects. The value of DT is either H (higher) or LS (lower, about the same).

Analyzing metrics. The review process package includes interpretation tables, defined in the context of the GQM, to help reviewers analyze these metrics. Table A shows how the RPC, RPD, ET, and DT metrics are used in measuring against the review process-measurement goals identified to determine a project's quality. For example, in row 1 the project score is 5, indicating a high-quality project. This score holds because we have done a good review and found few old or new problems. In row 4, the project score is 1, which indicates low quality. This score holds because we have done a good review and found more than the average number of new and old problems. In rows 5 and 6, a score of 0 indicates that we cannot make any conclusions because we have not done a good review.

These metrics provide managers with real-time feedback about a current project, without the need to wait for additional defect data to be collected. All the data necessary to evaluate the quality of a phase is available at the end of that phase.

Sample use of PCE. Phase-containment effectiveness is a key metric to quantify and track the improvement goal. You want to reach a value of 100 percent — the review is totally effective in finding all existing problems, assuming that some problems exist in the deliverable reviewed.

TABLE B
NUMBER OF DEFECTS INTRODUCED IN CONSTRUCTIVE PHASES

Phase	Number of errors	Number of defects
Requirements specification	5	0
Requirements model	12	1
Architectural model	11	4
Pseudocode	39	33
Code	10	10

In this example, the data comes from reviews done according to the review process package, augmented with testing and preliminary operation data.

We conducted reviews at the end of requirements specification, requirements modeling, architectural modeling, pseudocoding, and coding. As Table B shows, we found some errors during the review and they were fixed. Reviews of subsequent deliverables and testing, however, uncovered 48 defects that had escaped detection during review, listed in Table B in the constructive phase they were traced back to.

Using this data, you can derive phase-containment effectiveness for reviews done during each phase:

♦ Requirements specification review = 5/(5 + 0) = 100.00%

♦ Requirements model review = 12/(12 + 1) = 92.31%

♦ Architectural model review = 11/(11 + 4) = 73.33%

♦ Pseudocode review = 39/(39 + 33) = 54.17%

♦ Code review = 10/(10 + 10) = 50.00%

These metrics indicate that pseudocode and code reviews had relatively low containment values. Perhaps the reviewers need more training or the checklists need updating. In addition, the project participants should analyze the specific errors and defects using Pareto charts to determine their process-related causes and ensure that the process gets changed.[3] This should help avoid the introduction of such faults in the future.

REFERENCES

1. D.P. Freeman and G.M. Weinberg, *Walkthroughs, Inspections, and Technical Reviews*, Little Brown, New York, 1982.
2. D.L. Parnas and D.M. Weiss, "Active Design Reviews: Principles and Practices," *Proc. 8th Int'l Conf. Software Eng.*, IEEE CS Press, Los Alamitos, Calif., 1985, pp. 215-222.

from the organization's progress.

7. Analyze data from using the process package, evaluate the practices, and improve the process package. Proceed to step 1 and, armed with the recorded, structured experience gained from this and previous cycles, start the cycle again. Package this experience to make it accessible to others involved in creating process packages.

MOTOROLA'S EXPERIENCE

In 1988, Motorola's Metrics Working Group[3] was formed to develop and deploy, among other items, a process package for formal software reviews. The members of the Metrics Working Group were selected to represent Motorola business units whose goal is to champion measurement-based process improvement. It was to be part of a broader Software Engineering Technology Steering Committee and funded by Corporate Software Research and Development.

THE PILOT PROJECTS WERE MOSTLY ENHANCEMENT PROJECTS OF RELATIVELY SHORT DURATION.

The Metrics Working Group is similar to a Software Engineering Process Group, as later defined by the Software Engineering Institute. It is a volunteer group whose focus is process engineering and measurement, as opposed to an organization with a budget and head count. Individual Motorola business units have their own process and metrics working groups; if an organization does not have one, it is encouraged to create one.

Selecting a process. By applying the first two steps of the Software Engineering Improvement Paradigm, the group identified a set of improvement goals, one of which was to improve the software-review process. For several reasons, the group chose this process as the one with the highest potential payoff:

- It is an effective marriage of process and measurement.
- It covers the entire life cycle, so it provides feedback to all processes and methods and introduces the approach to every part of the organization.
- It is the most critical aspect of product evaluation, yet it was not being used widely in 1988.
- It helps find problems early.
- It provides critical defect baselines.
- It is a good first step for integrating other process packages. A review package can be instantiated for each review along the development path: requirements, design, code, and test script.

The formal software-review process, described in the box on pp. 72-73, was the first area in which we implemented the concepts embodied in the Software Engineering Improvement Paradigm and the process package.

Creating documents. After applying step 3, the Metrics Working Group drafted seven documents that became part of the review package, each targeted to a specific audience.

- *Overview* targets everyone. It lists the process-package documents and their corresponding audiences.
- *QIP* explains the Quality Improvement Paradigm to corporate-level managers.
- *Managers* tells software managers what to expect when they use the review package.
- *SQA* describes to software quality-assurance personnel how to use the review package.
- *GQM* explains to software managers and SQA personnel how to apply GQM to the review process and defines review metrics and how to use them.
- *Definition* describes in detail to software managers, SQA personnel, and developers how to implement a formal, technical review. It includes four forms designed to document the outcome of a specific formal software review.
- *Experience* gives corporate-level managers, software managers, and SQA personnel sample results and lessons learned in using the review package on pilot projects.

Selecting pilot projects. Once the initial versions of these documents had been created, the group selected a small set of pilot projects within one business unit (step 4). They chose mostly small enhancement projects of (relatively) short duration so that results would be available as soon as possible. The engineers and managers tailored the reviews over time and adapted the process to their needs. Their input, in turn, was used to enhance and evolve the initial review package. Acceptance of the review package was good, so it was generalized to apply to more projects.

The business unit's representative to the Metrics Working Group carefully monitored the use of the review package in the pilot projects. The group documented these lessons in the Experience document, and the package evolved over time to address the lessons learned on the pilots.

Training and consulting. As the review package was being implemented on pilot projects, the Metrics Working Group developed a one-day workshop that explained how to implement and measure software reviews (step 5). The first workshop was developed and taught by the authors of the review package to cover the mechanics of conducting reviews, in the context of the review package. The course covered technical and interpersonal communication issues.

Once the technical content stabilized, the course was transferred to Motorola's training organization, Motorola University, where it is now available to all Motorola engineers. It is not required for a group project that conducts software reviews. However, several training road maps include it as a recommended course.

In the last five years, we've offered this workshop to all development groups that want to use the review package to conduct formal reviews. If the project manager so requests, this training is followed by expert consulting, to ensure effective implementation of the ideas presented in the workshop.

Deploying a package. Over the next three years, the review package was deployed in several business units (step 6). This took about one person-year of tailoring and deployment work, primarily by the Metrics Working Group.

Package use was concentrated in smaller projects in business units where managers and developers had been trained and received follow-up consulting. Also, having an active champion to consult on how to use the package promoted wider use.

Evaluating a package. After the package had been in use about three years the group conducted a survey of about 100 engineers and managers across the company to determine how often the review package was being used, how it had been tailored, and what improvements were necessary (step 7).

The survey indicated that the review package was successful: 90 percent of projects within the business units affected conducted formal software reviews, and 67 percent of respondents said they used the review package.

However, 74 percent of respondents said they had had to tailor the process package. The items they changed most were the forms provided to document the review process. We did (and still do) encourage such tailoring, but wanted to identify what changes were done by what types of projects, so we could provide criteria for tailoring.

The items that did seem to work well were data-collection and error-tracking forms, reviewer sign-off, and the guidelines for whether or not to hold a review meeting.

The items that did not seem to work well are the assignment of roles to reviewers, the metrics charts used for data analysis and feedback, and the guidelines for implementing the roles assigned to reviewers. We are addressing these shortcomings through additional training and by creating local procedures.

The survey revealed that the primary inhibitors to use are the lack of appropriate resources, the lack of guidelines for how to apply the package to very small projects, and the need to streamline processes.

TABLE 1
PERCENTAGE OF FAULTS FOUND IN FORMAL REVIEW

Deliverable	Release 1	2	3	4
System functional specification	85	80	72	80
Software functional specification	78	67	80	70
Detailed design	49	78	64	81
Code	32	25	37	44+*

* = *Release 4 is not yet complete.*

LESSONS LEARNED

The lessons we learned hint at what we can expect as we deploy other process packages and what we must do to ensure that the tailoring and transferring of software-engineering technology is done effectively. However, some of the details are specific to reviews (such as the need to evaluate the reviewers' preparedness before a review meeting).

♦ *Don't underestimate the importance of champions.* Involving business units that will use the process package as you develop it not only ensures its acceptance, but facilitates the transfer process. The Metrics Working Group participants who helped tailor the review package became its champions within their business units. Business units that did not have representatives in the working group did not reap the benefits of this technology as readily.

♦ *Don't skimp on training.* We quickly realized that the one hour of training we initially offered to pilot projects was insufficient. As a result, we developed a one-day workshop and made it the first step in deploying the review package. We also conducted train-the-trainer sessions, to speed deployment in parts of Motorola that received many requests for training. We also found it was critical to follow up with expert consulting, which we discovered helped smooth the initiation of the formal review process.

♦ *Be prepared to be specific.* Once the developers understood the review process, they asked for more concrete guidelines. They wanted to know what role (leader, recorder, presenter, designer, and so on) each review participant should take, specific criteria for determining when they should not proceed to conduct a review meeting (due to lack of preparation, for example), and what to do with the results. To develop these role guidelines, we referred to the objective of each review type. For example, reading the requirements document from a tester's perspective assumes the reader is trying to understand if there is sufficient information to develop tests for each requirement. To develop other quantitative decision guidelines and criteria, we relied on data collected from reviews.

♦ *Preparation is key.* We found that the most important factor in predicting a review's effectiveness is how prepared the reviewers are when a review meeting starts. Review leaders asked for indicators to determine reviewer readiness, so we incorporated a form that asked reviewers to indicate the time they spent preparing for a review, and we tracked the number of errors found before and during a review meeting. We also found that review leaders were initially hesitant to issue a no-go decision to hold a meeting, even if the reviewers were ill-prepared or many errors were found. The consultants helped mitigate this tendency.

A SURVEY DONE THREE YEARS AFTER DEPLOYMENT INDICATED SUCCESS.

♦ *Data collection and analysis must be tailored.* The reviewers requested classification schemes to help them record defects and analyze the data for use in process improvement. We did develop classification schemes but found that they must evolve over time and are highly dependent on the type of project and product. The classification schemes provided valuable feedback to help us standardize and improve metrics collection, analysis, and reporting.

♦ *Formal reviews do improve quality.* When the review package was deployed, some small projects were not conducting any reviews at all, relying on testing to find faults. Formal reviews helped find and fix faults early, as the data from four successive enhancements of an internal project indicates. The data in Table 1 shows the percentage of faults found in each phase, in the early stages of deploying the review package. Note that unit and integration test found most of the faults escaping from these reviews. Reviews during the detailed design and code review phases show the biggest improvement.

Motorola's culture is such that business unit managers decide what process and technology will be used within their unit. Although senior management sets the quality-improvement goal, and the Metrics Working Group recommends formal reviews, the use of the process package is not mandatory. Data like that in Table 1 is far more effective than any mandate.

It is not easy to tailor and transfer software-engineering technology. To change the culture of the business unit so that it will accept new technology, you must employ champions and package information appropriately.

Using the Software Engineering Improvement Paradigm will help identify the process packages that should be developed first. Then, when you enjoy success on some pilot projects and publicize that success, new projects will sign up.

Many projects and locations across the company now use versions of the review package, and we have since created a testing package.

We believe our evolutionary, feedback approach has three main strengths:

♦ It provides quantitative guidelines that encourage the achievement of quality and productivity goals.

♦ It supports the development of a corporate memory because it integrates quantitative measurement.

♦ It provides a way to improve and tailor technology and process through data analysis.

The work done on the reviews and testing packages has evolved into an initial Best Practices and Technology Transfer Program within Motorola, which uses internal and external benchmarks and metrics to identify and promote effective, high-payoff practices to produce quality software. Motorola has also used benchmarking to establish aggressive improvement goals and metrics in software process, quality, cycle time, development technology, and customer satisfaction.

Building on the work done on the review package, Motorola business units have started to adopt, tailor, and evolve Michael Fagan's inspections-based improvement process,[4] resulting in further improvements in software quality and productivity. Motorola has begun to use education and skills training for senior and middle management as a way to enlist improvement champions across the entire corporation.

These mechanisms, coupled with the vision provided by a senior executive program, whose mission is to accelerate the pace of software improvement, are leveraging our technology-transfer initiative to bring about change much more rapidly. ♦

ACKNOWLEDGMENTS

We thank the many Motorola employees who participated in the Metrics Working Group over time, especially Mike Burke, Ann Miller, Ken Biss, and David Yen. Their work was a significant factor in ensuring the successful tailoring and deployment of the review package.

REFERENCES

1. V.R. Basili, "Software Development: A Paradigm for the Future," *Proc. Compsac*, IEEE CS Press, Los Alamitos, Calif., 1989, pp. 471-485.
2. V.R. Basili and D.H. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Trans. Software Eng.*, Nov. 1984, pp. 728-738.
3. M.K. Daskalantonakis, "A Practical View of Software Measurement and Implementation Experiences within Motorola," *IEEE Trans. Software Eng.*, Nov. 1992, pp. 998-1010.
4. M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, No. 3, 1976, pp. 182-211.

Victor R. Basili is a professor of computer science at the Institute for Advanced Computer Studies at the University of Maryland at College Park. One of the founders of the Software Engineering Laboratory, his interests include quantitative approaches for software management, engineering, and quality assurance. He serves on the editorial board of *Journal of Systems and Software*.

Basili received a BS in mathematics from Fordham College, an MS in mathematics from Syracuse University, and a PhD in computer science from the University of Texas at Austin. He is an IEEE fellow and a member of the IEEE Computer Society.

Michael K. Daskalantonakis is a lead engineer in Motorola's Cellular Infrastructure Group. His interests include software process engineering and measurement, technology transfer, software reviews, and project management.

Daskalantonakis received a BS in computer engineering from the University of Patras, Greece, and an MS in computer science from the University of Maryland at College Park. He is a member of the IEEE and ACM.

Robert H. Yacobellis is a senior member of the technical staff and manager of corporate software process engineering at Motorola. He serves on Motorola's Software Engineering Technology Steering Committee, Corporate Quality Council, Engineering Council, and Science Advisory Board Associates. His interests include software quality and metrics and Japanese software practices.

Yacobellis received an MBA and a PhD in information sciences, both from the University of Chicago.

Address questions about this article to Basili at Computer Science Dept., University of Maryland, College Park, Md. 20742; basili@cs.umd.edu or to Daskalantonakis at Motorola, 1501 W. Shure Dr., IL27/Rm 1315, Arlington Heights, IL 60004; dask@cig.mot.com.