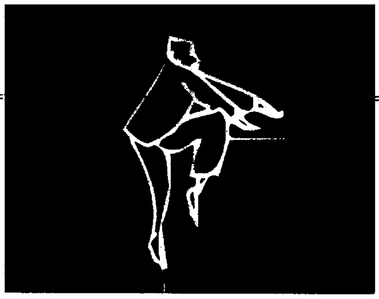


Key Lessons In Achieving Widespread Inspection Use

ROBERT B. GRADY AND TOM VAN SLACK,
Hewlett-Packard

◆ *HP has distilled its experience in promoting inspections into a model of how technology adoption occurs and a metric of where it stands. Its managers know when and how to accelerate efforts to adopt inspections and other best practices. Experience has shown that the return on investment in technology-adoption efforts can be huge.*

Buying a house is one of the biggest decisions you ever make. When Bob and Jan bought their house, they worried about a lot of things, including the heating system, which used hot water instead of forced air. They had been told that some such heating systems used pipes that had to be replaced after 15 years. Although they were told the pipes in their house were OK, they decided to have the heating system inspected anyway. It turned out the house's pipes were leaking and had to be replaced. The $25 inspection fee was the best investment they ever made. The sellers had to buy a new heating system that cost more than 100 times the inspection cost.

Most software inspections won't yield a return on investment of 100 to 1. Depending on your business though, some can, and our data suggests that you can expect a yield of about 10 to 1.

This is the story of software-inspection adoption over time in Hewlett-Packard and the insights we have gained.

Inspection technology was adopted across our entire company in stages. Figure 1 shows three versions of an S-shaped curve of technology adoption that researchers have shown typically occurs.[1] The left-most curve shows rapid adoption and the fastest ROI. The other curves show much slower adoption rates. The figure illustrates that the adoption rate of a new practice can vary significantly. It also sug-

0740-7459/94/$04.00 © 1994 IEEE

gests four recognizable stages, which we defined for this article: experimental, initial guidelines, widespread belief and adoption, and standardization. HP's inspections program has progressed through three of these stages.

In this article, we address several key questions: What characterizes the four stages? What were the most important lessons learned? What situations led to failure or success? And most important, how can we apply what we learned to speed the adoption of other proven practices?

The lessons we relate here will give many of you who are directly or indirectly responsible for software process improvement more confidence that inspections apply to all software-development organizations.

INSPECTION INFLUENCES

It is easy to identify three major influences on HP software inspections. The first of these is historical. All engineers do reviews of some sort. In HP two of the more obvious hardware-related reviews were for hardware designs and printed circuit boards, both motivated by the need to reduce costly cycle times that also delayed time to market. The first HP software reviews — walkthroughs, really — were modeled on hardware reviews.

Then in 1976 Michael Fagan published his very influential "Design and Code Inspections to Reduce Errors in Program Development,"[2] in which he introduced the term *software inspections*. For the first time, Fagan presented data that helped managers better visualize inspection results. He also described the inspection process well enough that commercial classes soon followed. HP's first internal classes were created in the late '70s and widely taught by the early '80s.

The third influence was Tom Gilb. He extended Fagan's inspection process in several important ways that represented a timely philosophical match with HP thinking. First, he was a strong proponent of inspecting early life-cycle artifacts more than code. Second, he extended Fagan's application of metrics to include measures of the inspection process itself. Finally, he described how to use these measures to motivate process improvement.[3] In 1990 HP combined these improvements and our accumulated experience with Fagan's methods to revise the inspection classes that are internally taught today.

ADOPTION RATES VARY WIDELY, USUALLY PROGRESSING THROUGH FOUR STAGES.

The current HP inspection process is summarized in the box on p. 48. It is the result of more than 15 years of experience and improvements. The roles, steps, and results are well-understood and effective. The process works. By mapping the evolution of the program onto the four stages in Figure 1, we can better understand how adoption grew and how the program's success can be repeated by other process-improvement programs.

EXPERIMENTAL (1976-1982)

In describing HP's early inspections experiences, we can appreciate how hard it is for archaeologists to reconstruct early history without any written records. In this case, we can at least summarize the memories of some of the people who were doing software then, but their recollections are bound to be fuzzy and biased.

Fagan's article provided a common point of departure for different HP groups. What were once isolated reviews initiated by former hardware engineers and project managers now became more organized efforts supported by managers in quality assurance and, occasionally, research and development. During this time, only a few HP divisions had quality-department staff with software backgrounds. Fagan's article gave even those who did not have software backgrounds something specific to work on with software people.

At this stage, our inspection process

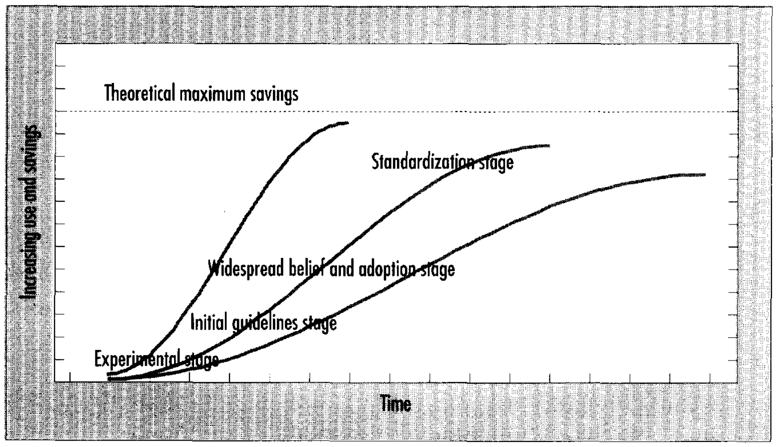


Figure 1. Three versions of an S-shaped curve of technology adoption. The leftmost curve shows rapid adoption and the fastest ROI. The other curves show much slower adoption rates. We define four recognizable adoption stages: experimental, initial guidelines, widespread belief and adoption, and standardization.

CURRENT HP INSPECTION PROCESS

The HP inspection process involves conducting a formal review of a document by a document owner and a team of peers. The review is led by a process chairperson, the moderator. The primary goal of the inspection process is to help the document owner and organization develop a high-quality product cost-effectively. Inspections ensure that documents are clear, self-explanatory, correct, and consistent with all related documents.

Five roles. There are five defined roles. Sometimes one person acts in more than one role.

♦ *Inspectors:* Find items (errors, omissions, inconsistencies, and areas of confusion) in a target document and issues (items beyond the scope of the inspection team).

♦ *Scribe:* Records items and issues found both before and during a logging meeting.

● *Owner:* Owns a document; identifies and fixes defects.

♦ *Moderator:* Manages the process; facilitates the inspection; reports statistics to the chief moderator; reports inspection's status and logged issues to management.

♦ *Chief Moderator:* Owns the inspection process; gathers and reports statistics across all inspections; drives inspection process improvements; serves as focal point for changes to inspections standards and checklists.

Some divisions also continue to use the *reader* role (a person who paraphrases a document during a logging meeting), particularly for complex code documents.

Seven steps. Table A shows the seven distinct steps in the HP inspection process, what happens, the people involved, and the time spent. This well-defined process represents a transformation of HP's earlier reviews and walkthroughs to an engineering process. Figure A shows how the process outputs are used to improve our products, inspections process, and development processes.

Occasionally the inspections process varies. For example, some divisions hold more formal meetings than others. In general, HP's culture leads to divisional autonomy and less formality than in many other companies. One aspect that the current HP inspection process emphasizes, though, is that informality must not extend to the recording and tracking of meeting results.

The process step that varies the most is the cause/prevention step. Few divisions have successfully done this step before rework. The most successful adaptors of this step have the engineers who fix the defects enter causal information and short recommended actions when fixes are done. Then a separate failure-analysis activity takes data both from multiple inspections and from other test results to analyze and brainstorm solutions to frequently occurring or expensive problems.

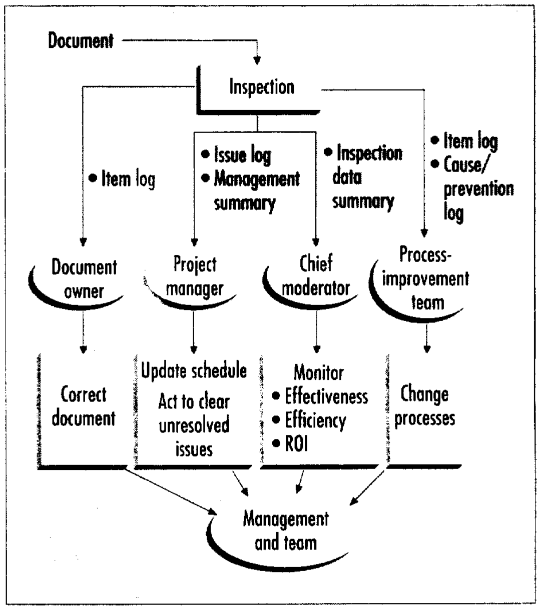


Figure A. *How HP uses the results from inspections.*

TABLE A
PROCESS STEPS AND CHARACTERISTICS OF HP SOFTWARE INSPECTION

	Planning	Kickoff	Preparation	Logging meeting	Cause/prevention	Rework	Follow-up
What	Plan inspection and create packet	Brief team	Find items and issues	Find and log items and issues	Brainstorm cause(s) and recommend	Verify and fix defects	Release document
Who	Moderator and owner	Team	Team	Team	Team	Owner and moderator	Moderator and owner
Time	2 hrs.	1/2 hr.	2 hr.	1 1/2 hrs.	1/2 hr.	???	1/2 hr.

used Fagan's checklist as a starting point and had code readers look at code from different perspectives. The emphasis was on code inspections.

Some inspections compared their metrics with Fagan's published metrics for engineering times, defect-finding rates, and defect types. Few matched Fagan's data closely enough for people to feel very positive about their results. Some of these start-ups failed. Many groups had little more than Fagan's article to go on, and for some projects this was not enough to reshape the team's dynamics, which sometimes involved strong egos and tendencies to criticize. Word of failures often spread faster than word of successes.

Some groups who believed in inspections persevered and tuned the process. Those who survived the start-up period became advocates for better company support. These groups found that better training, documentation, and a supportive atmosphere were necessary for successful inspections.

The experimental stage taught us that

♦ The initial use of new processes, methods, or tools depends heavily on visionary people who can look at someone else's results and see how they can be applied locally. This often means having the right person in the right place at the right time.

♦ If a division wants to lead in the adoption of best practices, it must foster visionary attempts and not penalize failures. A risk-averse environment does not grow leaders.

♦ Early success is fragile. Establishing a supporting infrastructure can make the difference between success and failure, and it can be difficult to convince people to try again.

INITIAL GUIDELINES (1983-1988)

This period marked a transition in HP not only for inspections but also for the use of many other software best practices. It is easier for us to reconstruct events in this period because by now HP had formed its Software Engineering Lab and Software Engineering Training. SEL started its Software Engineering Productivity Conferences in 1984, and SET started a newsletter, *SET News*. Their written records helped us reconstruct this period's key events and lessons.

In 1983, HP piloted an evolved version of an internal software-inspections class. SET was created to propagate such practices through training. It expanded class availability, and in 1984 more than 150 engineers took the class, and SET was offering train-the-trainer sessions. The SEPC proceedings helped us understand the outcomes from this training.

GROUPS SURVIVING THE START-UP PERIOD BECAME ADVOCATES FOR BETTER SUPPORT.

A paper presented at the first SEPC showed how one group had adapted Fagan's checklists to a specific HP division. By 1985 there was already a shift to reporting results in measurable terms, most commonly in defects found per hour that ranged from 0.24 up to 5.18. Most of the variation was probably caused by varying definitions and different software types and business applications. These variations did not matter much, because the repeated success stories at the conferences were strong reinforcing arguments that inspections did work, and they played a key role in building inspections momentum.

Besides centralized training and the SEPCs, three other key events occurred during this time. The first was a two-day course on the role of managers in software quality that every R&D manager was to take over the next one-and-a-half years. Taught mostly by their peers, this course was particularly important, because it exposed R&D managers with hardware backgrounds to the software-development process, which most were not familiar with. It gave them a much better understanding of what software developers do and led to stronger management support for process improvement.

The second key event was the creation of a productivity-manager position in most HP divisions. The primary responsibility of these new managers was to speed the adoption of best practices within a division. These managers formed a basic infrastructure to actively support division R&D improvements.

Finally, in 1986 our company president announced a Software 10X Improvement Program. This program challenged all HP R&D product labs to improve two key quality metrics by a factor of 10 — 12-month postrelease defect density and open critical and serious defects.

Combined with documented success stories, an infrastructure that provided a divisional conduit to support change, and freshly fanned management commitment, the 10X challenge was an additional incentive to adopt best practices quickly. A 1987 internal survey of quality managers showed that inspections were clearly their first choice to help them achieve the 10X goals.

This was an exciting time for HP software developers. During this stage we learned that

♦ Communicating successes effectively speeds both adoption and the improvement of the best practices themselves.

♦ Clearly defining who is responsible for process improvement speeds the adoption of best practices.

♦ Management training contributes to strong, sustained sponsorship.

♦ A high-level, compelling vision, like 10X goals that are directly tied to business challenges, helps ensure strong management sponsorship.

♦ Readily available training is nec-

RETROSPECTIVE FROM INITIAL GUIDELINES STAGE

Let's look at the use of inspections by two different project teams within one division. Both team's firmware R&D engineers attended the 1984 version of the software-inspections course, which taught the Fagan method. It covered the reader role, discussed problems that can arise in inspection meetings, and how to achieve group consensus on defects. But it included little about how to use metrics to control and improve inspections, and there was not yet an infrastructure in place to support teams trying to use inspections.

Team 1 used the method to inspect code, and it helped them find defects. However, they felt that they weren't getting the ROI they should, but didn't know why (they didn't collect process metrics). Gradually they stopped using inspections. For one thing, the project manager thought they were too slow and people-intensive. For another thing, the team felt that inspecting noncode documents (requirements, specifications, designs, and test plans) took too long and didn't uncover enough major defects.

Team 2 used the method for code inspections and, like Team 1, tried to use the method for noncode documents. They experienced similar results, but instead of dropping inspections they set about improving the process. First they eliminated the reader, who used to paraphrase the document, and instead had the moderator step the team through the document. They adjusted the speed to the potential for defects. This helped the team focus the meeting on the document sections that had the most problems, and they achieved a better defect-finding rate.

Team 2 also experienced unproductive meeting discussions. To solve this, they eliminated discussion of whether or not a potential defect was really a defect. Instead, they had the inspector log any problem. After the meeting, the document author reviewed the log to decide if any follow-up discussion was needed. This also sped up the meeting and increased the number of potential major defects reported.

Finally, Team 2 had inspectors report their preparation time and the number of defects they found at the start of each inspection meeting. This helped ensure that every inspector spent enough time reviewing the document before the meeting. All these process changes resulted in more successful noncode inspections.

Despite the fact that both teams were in the same division, only one took the initiative to adjust the process. This happened in many parts of HP. Training by itself was not enough to ensure success, and an unsuccessful experience made it difficult to get groups like Team 1 to try inspections again.

essary to speed technology adoption, but it is not sufficient to sustain use (see the box above).

WIDESPREAD BELIEF AND ADOPTION (1989-1993)

By 1989 the 10X program was far enough along that divisions were reporting and corporate was summarizing data reasonably consistently. While some progress was apparent, pressure had increased on the divisions to show how they planned to achieve the goals. An internal corporate-wide survey showed that every HP division was doing some form of inspection, although only 19 percent regularly did software-design reviews. The survey also showed strong confidence in inspections — more than 30 divisions picked inspections as a key practice in their effort to achieve their 10X goals. While some divisions were working hard to implement widespread inspections, only five had plans for inspections training. It seemed clear that renewed corporate consulting efforts could help.

HP kicked off a new initiative specifically designed to

♦ improve the efficiency and effectiveness of inspections by implementing an inspection-moderator certification program, revising the training course, and instituting a company-wide technology-transfer plan

♦ improve R&D management awareness and commitment by developing a business case for inspections, marketing inspections, and monitoring division performance.

Four people from Corporate Quality worked part-time in close partnership with key practitioners to meet these goals. In early 1990, an internal inspections-practitioner workshop was held, and an outside consultant was hired to incorporate the best practices summarized there into an existing class. This formed the basis of a Moderator Training Class that soon followed. Meanwhile, an HP person created a "management pitch kit" on the business case for inspections.

These efforts culminated in a blitz campaign to promote inspections and the new training class across HP. Fagan and Gilb spoke at that summer's SEPC, and some of the key practitioner-workshop participants contributed four excellent papers. The blitz campaign created the awareness and management commitment to upgrade people's skills. This new commitment created demand for corporate, group, and divisional productivity and quality-assurance people.

HP responded by assigning both a full-time corporate inspections program manager and a full-time training specialist for one of the largest HP Groups (at HP, a Group is a collection of divisions in related businesses and may have more than 1,000 software engineers). This Group and corporate formed a partnership: Corporate provided guidance and funding. The Group championed improvements, adapted the moderator-training class for learning through practice, and modernized the inspections class. The Group felt that having these classes delivered externally was too expensive, so they planned to have their own people teach them. As Figure 2 shows, by early 1991, significant numbers of engineers were receiving the training.

These results did not come easily. The corporate approach to certifying moderators was based on the premise that there is one "best" way to do inspections and all the group had to do was push people toward it. But HP's divisions are independent and did not respond well to standardization. They didn't want to just be told what wasn't right, particularly when they saw their inspections working. They wanted flexibility, but they also wanted help in improving their processes. There was a constant struggle to hold management to their commitment to continuous improvement. What the divisions wanted was consulting support tailored to the Figure 1 adoption phase they were in.

Meanwhile, the training specialist was also having problems. One of the first classes "exploded" when a manager set up the class, then gave the engineers mixed signals about its importance versus other activities. Other classes included attendees who had tried inspections and not found them useful, and no one had told them how the methods in the new class might correct earlier problems. The training specialist came to the same conclusion as the corporate program manager: Consulting was needed to assess where a division was and to create an environment for success *before* training was done.

The consulting approach as it evolved in HP was adaptable to the needs of each division, which usually fell into three broad categories:

♦ Organizational commitment. Somebody — a key manager or key engineers — must be convinced that inspections could and should be done sooner instead of later.

DIVISIONS WANTED SUPPORT TAILORED TO THEIR ADOPTION STAGES.

♦ Advice and training on improving inspections. Often the local inspections champion was using a process that wasn't good enough for the division to feel that it was indispensable.

♦ Help in adapting the updated inspections approach to the division.

We then developed a consulting model structured to match and meet customer needs, provide appropriate training, and offer follow-up until the customer succeeds. It has five basic steps:

♦ Define the organizational business objective for doing inspections.

♦ Evaluate and influence the organization's readiness to do inspections.

♦ Create an infrastructure for success by identifying a local interested person as chief moderator (who will also act as a champion).

♦ Benchmark the current process.

♦ Adjust the current process, train people, and consult to ensure success.

We knew from experience that the number of people trained is only a weak predictor of successful adoption. We wanted a stronger predictor, so we created a survey to determine the percentage of inspection penetration and effectiveness across HP. We defined a division's degree of penetration as the percentage of projects that held four or more inspections during a project's life, because one division had had great success when it asked teams to do four inspections before judging if inspections were worth the effort.

We had also consulted more with larger labs, and we wanted to see if they had higher penetration, so we arbitrarily broke the data into four different size groups. Figure 3 shows both the average penetration for different lab sizes and the percentage of the total HP population in each category. For example, labs with more than 75 software engineers represent more than 50 percent of HP's total software engineers, and they did four or more

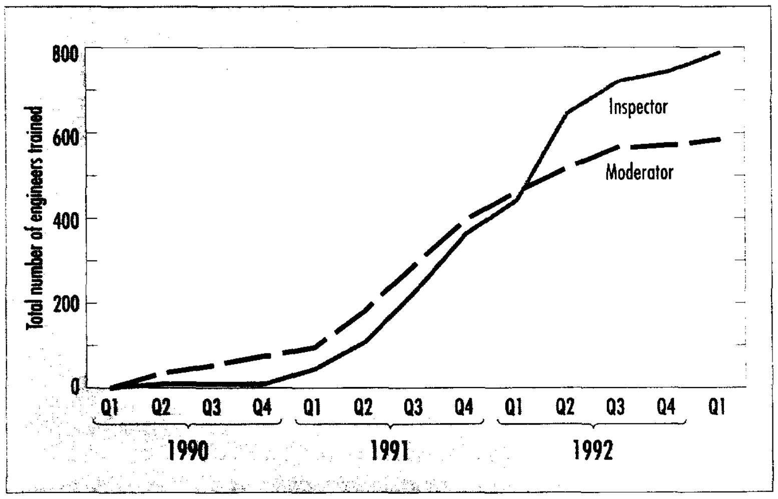


Figure 2. Cumulative inspection training. By early 1991, significant numbers of engineers were receiving training.

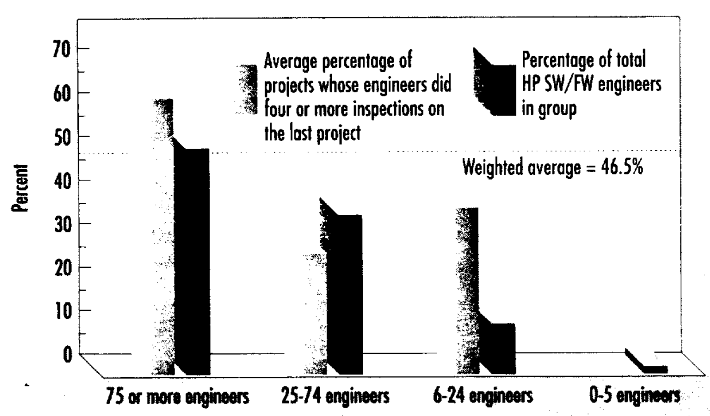

70
60
50
40
30
20
10
0
Percent
Average percentage of projects whose engineers did four or more inspections on the last project
Percentage of total HP SW/FW engineers in group
Weighted average = 46.5%
75 or more engineers
25-74 engineers
6-24 engineers
0-5 engineers

Figure 3. 1993 inspections penetration by number of division software and firmware developers. Labs with more than 75 software engineers represent more than 50 percent of HP's total software engineers, and they did four or more inspections on 63 percent of their projects in 1993.

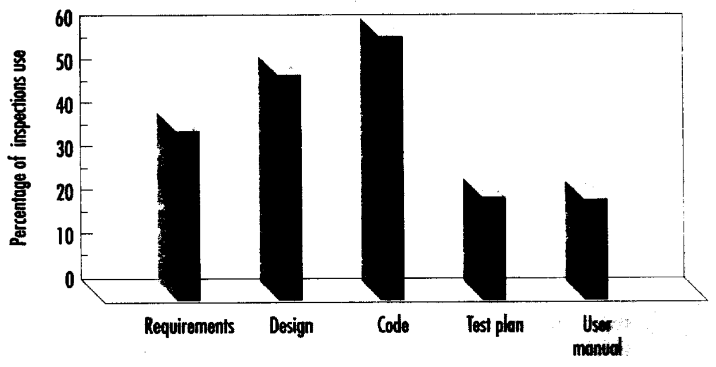


Figure 4. 1993 inspections use by document type in divisions with greater than 25 percent penetration. For this diagram, a project "used" inspections for a particular document if it inspected at least one document of that type.

engineers, and they did four or more inspections on 63 percent of their projects in 1993.

The 1989 survey showed that every division was doing inspections, but it didn't ask how many projects did them, or how many were done per project. The 1993 data in Figure 3 more clearly shows that more than half the projects were either not using inspections or were just starting to use them. Fortunately, this graph is a lagging indicator. Many people trained in 1992 and 1993 were working on projects that simply hadn't completed four inspections yet. Although the average penetration increased from 32 percent in 1992 to 46.5 percent in 1993, there is still much room for improvement.

The survey data does contain encouraging signs, however. Years of research show that the S-shaped diffusion curve takes off at about 10 to 25 percent adoption,[1] and more than 60 percent of the divisions responding had already achieved 25 percent penetration. Those who haven't are groups who can best benefit from continued consulting.

The survey also asked what documents are inspected. We strongly encourage the inspection of early documents, because these inspections help uncover defects that are very expensive if not found until later. We now analyze what percentage of different document types are inspected each year, but the 1993 survey asked only if projects inspected at least one noncode document of each type. For example, Figure 4 shows that 51 percent of the projects in divisions with greater than 25 percent penetration did at least one design inspection (and when all reporting divisions are included, the average is still 35 percent). This is much better than the 1989 average of 19 percent, and it suggests that recent training and consulting have helped increase early document inspections.

The survey clearly shows that inspection use is increasing, but the rate of increase varies. Because we are interested in how proven technologies such as inspections can be adopted faster, we looked more closely at one HP Group who has been very active during the last few years. This Group had a full-time person responsible for inspections and people who regularly both scheduled and did training. It also had high-level management support. One of the Group's key goals was to reduce rework, and using inspections was a major strategy to do this. Figure 5 shows the dramatic difference in the increased number of inspections done by this Group (74 percent) compared with the rest of HP (21 percent) from 1991 to 1992. We

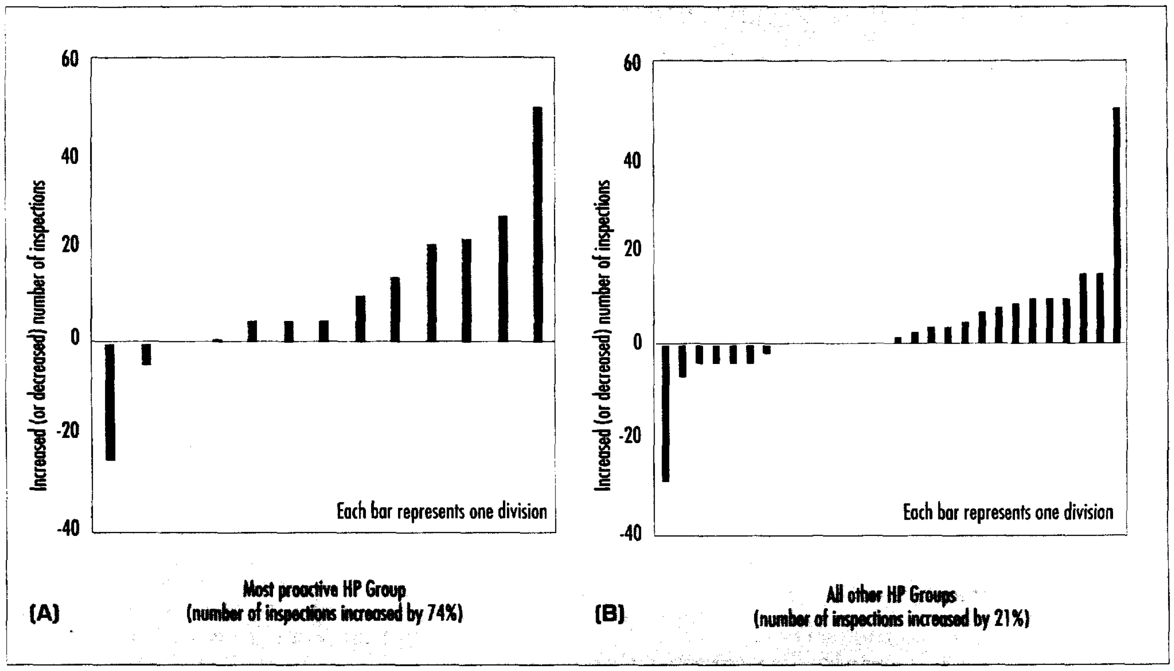


Figure 5. Increasing divisional use of inspections. (A) This Group had a full-time person responsible for inspections, people who regularly both scheduled and did training, and high-level management support. (B) Compared with the rest of HP, the increase in inspections from 1991 to 1992 for the proactive Group was dramatic.

also found that the most consistent difference between divisions that improved and divisions that remained unchanged or actually did fewer inspections was the presence or absence of a divisional chief moderator. Of the six divisions with the most growth in Figure 5a, every one had a chief moderator. It is clear that a good divisional infrastructure is as important as the Group and corporate ones.

Although we are still in the widespread belief and adoption stage, we have already learned that

♦ Training and process guidelines are not sufficient. Consulting with objectives tailored to divisions is also required. Find champions (for inspections, chief moderators) and sponsors. Work with divisions after training to ensure success.

♦ Few divisions can start with an optimal process. Learn the divisions' immediate wants and its long-term plans. Understand its processes and organizational constraints.

♦ Program-focused support makes a big difference in the adoption rate (Group-level people with full-time inspections responsibility, in this case).

♦ Good management process metrics are necessary to achieve and sustain widespread use and effectiveness (see the box on p. 54).

STANDARDIZATION

It's not clear that there ever will be a single "standard" HP inspection process, but this is not the issue. What matters is that every project use some variation of a standard process in an efficient, cost-effective way.

The critical leadership roles during all other stages are those of champion and sponsor, and the standardization stage is no different. The first part of our standardization plan is to continue to *proactively identify and support champions and sponsors*, who will provide the technical and organizational leadership necessary for each group to assume long-term process ownership and responsibility.

We must arm these champions and sponsors with strong, persuasive messages. The second part of the standardization plan is to *reinforce management awareness with a strong inspections business case.* Many success stories have been presented at the SEPCs. Some of these stories have gone beyond measuring inspection defect-finding rates to comparing them with defect-finding rates in systems test. Although the

TABLE 1
COST BENEFIT ANALYSIS OF DESIGN INSPECTIONS (ROI > 10 IN FIRST AND SUBSEQUENT YEARS)

Items	Costs	Benefits
Training	48 engineering hours $1,650 (= approx. 25 eng. hrs.)	
Start-up costs	96 engineering hours 0.5 month	
Reduced defect-finding time		1,759 engineering hours
Reduced time to market		1.8 months

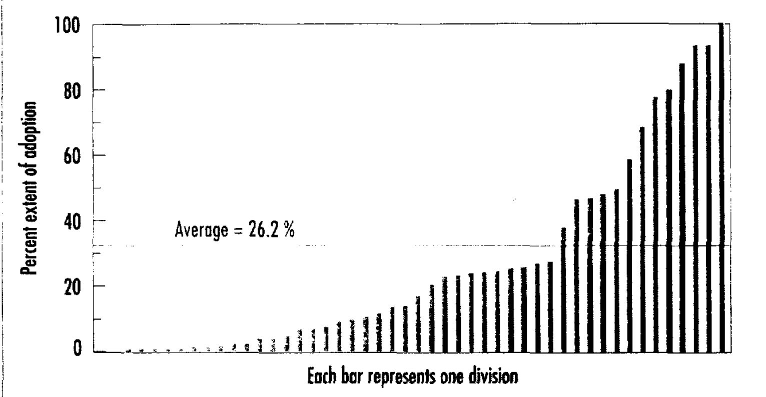


Figure 6. *Extent of inspections adoption, reflecting an increase from 20 percent adoption in 1992 to 26.2 percent adoption in 1993.*

RETROSPECTIVE FROM WIDESPREAD BELIEF AND ADOPTION STAGE

·The experiences in two divisions show how our thinking has gone beyond just implementing inspections to how inspections affect our products.

In the first division, code volume was growing rapidly. Fortunately, they had good historical labor and defect metrics, so the quality manager could make a strong case that future projects were in jeopardy unless they could substantially reduce testing effort and schedule by using inspections.

The division, which had a strong champion, also provided other incentives to adopt inspections. It conducted an exercise that had one team inspect a "clean" module written by one of its best engineers. The inspection uncovered three significant problems, so the engineers were convinced. And it was one of the first divisions to integrate real work products into its training class instead of canned examples and to train work groups together.

In three years, this division has done more than 180 inspections. The average time to find and fix major defects using inspections continues to be more than 10 times better than their old test methods, as reported by Dave Dickmann in a paper presented at an HP SEPC. They are getting more complex products to market far sooner than they would have using their old process.

The second division already enjoyed widespread use of inspections and it remained high even through several reorganizations. Inspections typically found 60 to 70 percent of the defects. However, a review of postrelease critical/serious defects showed no significant improvement in the division's products. To learn more, 13 project managers were asked about their backgrounds, their reasons for doing inspections, details about the inspections, what data was collected, and what follow-up occurred.[1]

What they learned was that a strong belief system had kept inspections alive, but the process was stagnant: It focused on code and other back-end work products; entry and exit criteria had relaxed; and there was seldom any attempt to use inspection data to change the process. This division learned that it must monitor the effectiveness of the inspection process by examining the quality of its products.

Questioning the process was a healthy exercise. The results created both motivation and direction for change. The key changes included a greater front-end emphasis and stronger metrics to drive process improvements and to monitor their inspections process.

REFERENCES

1. G. Shirey, "How Inspections Fail," *Software Practitioner*, May/Aug. 1993, pp 12-17.

rates vary from 4.4 times better up to 20 times better, they all make a strong business case.

Table 1 shows a cost-benefit analysis for design inspections, developed with well-documented HP and industry results. These numbers are for first-time application of design inspections on a six-person project to produce 50,000 lines of noncomment source statements. Because of space, we cannot detail the assumptions and calculations behind this analysis here, but we have published them elsewhere.[4] The ROI is a simple ratio of engineering time benefits divided by costs (1759/(48 + 25 + 96) = 10.4). Potential added revenue from faster time to market improves the ROI even more.

This is a much better return than many other R&D investments provide. In addition, there is little risk, and you needn't wait long for the initial investment to be paid off. Note that training and start-up costs occur only once per team, although there will be a small ongoing cost for a chief moderator. This type of analysis is important, because later adopters of new practices are more conservative and need stronger persuasion.

A software-organization's management must believe that inspections are important enough to develop them as a key part of their *core competence*, a set of skills that give a company a competitive edge. These skills represent collective learning that is "built through a process of continuous improvement and enhancement that may span a decade or longer."[5]

The third part of the standardization plan is to *continue building an infrastructure strong enough to achieve and hold software core competence*. At HP, the three layers of the infrastructure are corporate, group, and division. Their key roles are to train, consult, communicate, improve, and be responsible for the inspections process. We've seen that such an infrastructure maximizes the adoption rate and increases the odds that inspections are optimally done.

The fourth part of our standardiza-

DEFINING EXTENT-OF-ADOPTION METRIC

We created the extent-of-adoption measure to gauge company-wide progress against the three key program emphases: process maturity, depth of use, and breadth of use. The metric can range from 0 to 140, but we set all calculated values greater than 100 to a maximum value of 100. Briefly, the components of the equation

extent of adoption = inspection-process maturity
*(percentage of projects using inspections + weighted percentage of documents inspected) * constant

are as follows.

Inspection-process maturity. A constant from 1 to 14 based on a five-level model that details the practices early adopting divisions used to improve their peer-review processes. Model levels 1 and 2 are not really inspections, but informal peer desk reviews to formal walkthroughs. Level 3 is an industry-typical inspection process. Level 4 is a best-practice inspection process. And level 5 links formal inspections to a defect-prevention activity.

♦ *Level 1: Initial/ad hoc* (weight = 1): No established objective for doing reviews; individual peer review/desk-checking to informal walkthroughs done; no process documentation, training, or process infrastructure.

♦ *Level 2: Emerging* (weight = 3): Informal objective to find and remove defects; informal and formal walkthroughs done; process may be documented, some trained practitioners, and some awareness of need for infrastructure.

♦ *Level 3: Defined* (weight = 10): Stated objective to find and remove defects; formal inspections recommended on major projects; process documented, moderators trained, infrastructure in place, metrics collected; standards, templates, and checklists may not exist.

♦ *Level 4: Managed* (weight = 12): Stated objective to improve defect detection and removal earlier in the life cycle; formal inspections required on key documents for all major projects; process metrics used to improve process, defect cause information collected; standards, templates, and checklists exist for all key document types.

♦ *Level 5: Optimizing* (weight = 14): Stated objective to prevent defect introduction; organizational standard exists — and is adhered to — that defines which documents will be inspected under what conditions; defect-cause information is used in root-cause analysis to target changes to development process.

Percentage of projects using inspections. The percentage of projects that did four or more inspections on their last project. We picked four based on the experience of one division that had great success when it asked teams to do four inspections.

Weighted percentage of documents inspected. Several studies have measured the relative costs of fixing different types of defects, and our weights are based on their findings.[1] We then multiply these factors by the organization's percentage of projects using inspections for each document (we define a project as using inspection if it inspects at least one document of a particular type).

(5.7 * percentage of requirements + 2.5
* percentage of design + 2
* percentage of test + percentage of code)/11.2

We derived the percentage using the survey results in Figure 4 in the main text. For Figure 6 in the main text, because the survey only asked for the presence or absence of inspections for different document types on projects, the results overstate actual use. This measure has a factor of 11.2 so that its value also ranges from 0 to 100.

Constant = 0.05. Chosen to fit the extent-of-adoption metric into a range of 0 to 100 when the inspection process maturity factor is at a level 3 (weight of 10) and both the percentage of projects using and the weighted percentage of documents inspected are at their maximum values.

There is no scientific basis for assuming that a level 2 inspection process is three times better than a level 1 process or that a level 3 process is 10 times better, and so on. The maturity weights and their mathematical use in the extent-of-adoption metric are intended only to identify trends. As with all metrics created for one purpose, there is always the potential that some people might use this one for other purposes. To minimize the extent of potential distortions, we did a series of sanity checks that tested all combinations of the three factors at their high and low conditions. We then asked ourselves if the result seemed reasonable. To us, they did.

REFERENCES

1. R. Grady, *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, Englewood Cliffs, N.J. 1992.

tion plan is to *measure the extent of adoption*. We believe that consistently improving inspections use will visibly reduce defects and development times. We needed a measure that ties inspection results as closely as possible to such improvements. We created a measure consisting of three adoption components.

♦ *Depth* (percentage of projects using inspections). Some minimum number of inspections that signifies a project team seriously believes inspections are necessary (it is four now, but it may change).

♦ *Breadth* (weighted percentage of documents inspected). A weighting factor that accounts for the observation that inspecting early work products (requirements, designs) yields better returns than inspecting late ones (code).

♦ *Inspection-process maturity.* Not all inspections are equal. Today's inspections process is much more effective than Fagan's first method, so we add a simple weighting factor, as described in the box above.

So the formula we use to measure adoption, which we have found does separate the most effective divisions from the least effective is:

extent of adoption =
inspection-process maturity
* (percentage of projects using inspection
+ weighted percentage of documents
inspected) * constant

This is how we produced the data in Figure 6. Imperfect as this metric is, it gives a baseline against which to track year-to-year progress. For example, the metric reflects an increase from 20 percent adoption in 1992 to 26.2 percent adoption in 1993. Most important, it includes the three major aspects that our experience shows contribute to reduced costs and development time.

ESTIMATING SAVINGS FROM INSPECTIONS

We derived the 1993 estimated savings of $21.5 million using the formula

estimated $ savings/year = % total costs saved * rework % * efficiency factor (.4) * total engineering costs

Percentage of total costs saved. Percentage for a work-product component only. It peaks at 100%.

Rework percentage. An internal HP software-development cost model estimates total rework at 33 percent. It is broken down into the work-product components shown in Table A.

Efficiency factor. Capers Jones says this varies from 30 percent to 75 percent.[1] Assume 40 percent for 1993 because our average maturity level is still relatively low.

Total engineering cost. Assuming 3,500 R&D software engineers at a cost of $150,000 per engineering year, total cost is $525 million.

Maximum possible savings from inspections. Total engineering cost ($525 million) times the rework percentage (33 percent) times the efficiency factor (60 percent) equals $105 million. (Some of this will be saved through other engineering techniques, so while this is a theoretical maximum, the practical maximum will be somewhat less. Also, while inspections substantially reduce costs, they don't totally eliminate them. We use a 60 percent efficiency factor to simulate these combined effects).

Total savings. The sum of the savings from four major work products. Take the increase in the percentage of different types of inspections (for example, our 1993 survey showed design inspections had increased 33.8 percent), multiply it by how much of the total cost ($525 million) you assume the rework of a work product accounts for (11 percent in the case of design) and multiply that by an assumed 1993 efficiency factor of 40 percent. So 1993 design savings are

33.8% * (11% * $525M) * .4 = $7.81 million

These are very rough calculations, but they give us a way to translate our extent-of-adoption measure to company-wide savings. The estimated 1993 percentages came from the 1993 survey, which asked only about the presence or absence of different inspection types on projects, so the results overstate actual usage. The efficiency factor of only .4 at least somewhat makes up for this. Table A summarizes the savings estimate.

REFERENCES

1. C. Jones, *Programming Productivity*, McGraw-Hill, New York, 1986, p. 179.

TABLE A
ESTIMATED YEARLY SAVINGS ATTRIBUTABLE TO HP SOFTWARE INSPECTION

Work product	Estimated starting point	Estimated 1993	Percentage total cost saved	Rework pecentage	Estimated $ savings per year
Specification	1%	29.5%	28.5%	17%	$10,175,000
Design	1%	34.8%	33.8%	11%	$7,808,000
Code	5%	42.3%	37.3%	4%	$3,133,000
Test plan	1%	17.1%	16.1%	1%	$338,000
Total				33%	$21,454,000

TABLE 2
KEY CONTRIBUTORS TO SUCCESSFUL INSPECTION ADOPTION

	Experimental	Initial guidelines	Widespread belief and adoption
Business factors		Compelling vision (10X)	Recognition of need for core competence
Organizational readiness	Local support infrastructure	Company-wide infrastructure (productivity managers) Company -wide communications (SEPC)	Inspections infrastructure (company, group, divisional levels) and metrics Needs assessment — organizational plans, readiness, constraints Consulting to set stage for success
People factors	Visionary people (champions) No penalties for failures (sponsorship)	Management training (sponsors) Local adaption	Proactive identification of champions and sponsors

Software inspections have taught us a lot about technology adoption. Table 2 summarizes the key lessons we learned in each stage. Where is HP on its S-shaped curve? To answer that, we must first compute the theoretical maximum savings. We estimate that roughly one third of all HP software costs are rework, and inspections can save 60 percent of these costs. For a company the size of HP, a conservative potential savings then is $105 million *per year*. (The box above explains how we arrived at these rough calculations.) Next, estimate how much the inspections program is already saving. Using data from our 1993 survey, we separately estimate savings for each document type and add them, for an estimated savings in 1993 of $21.4 million. Note that when these savings are translated to a lab of 100 engineers, they yield a bonus of more than four engineers to work on things other than rework (in every lab across the entire company). It's no wonder that a US government study concluded that it took an average of 15 to 20 years for new software technology to get to the point that it could be popularized and disseminated.[6] It has taken HP more than 15 years to reach this point, and we feel we still have almost 80 percent of the benefits yet to gain!

It is too late to speed up the earlier phases for inspections, but it is not too late to apply what we have learned. HP has created a set of software initiatives to accelerate company-wide adoption of a small number of software best practices, including inspections. These initiatives provide timely help to divisional pilot projects for these best practices and the corporate-level infrastructure we learned was so key to speeding technology adoption.

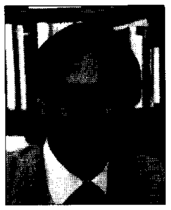
It is through experiences like those we describe here that we learn how to do things better. We now have a model for how technology adoption occurs, we know what to do to accelerate improvement, and we've been exposed to the typical problems we can expect. These experiences can be and are being applied to accelerate adoption of other best practices, and we've seen that the returns for such investments can be huge. ◆

ACKNOWLEDGMENTS

It is impossible to name everyone who played a major part in our program's success, but we would like to publicly thank the main contributors in recent years. These people helped customize and champion inspections at HP: Babs Brownyard, Neil Davies, Dave Dickmann, Clairmont Fraser, Bob Horenstein, Billy Knorp, Jean MacLeod, Debra Mallette, Rose Marchetti, Patsy Nicolazzo, Richard Peifer, Jerry Peltz, Steve Rodriguez, Jack Ward, David Wittall, and Dawn Yamine. In addition to some of them, others wrote some of the excellent papers that helped to sustain HP's interest and beliefs: Lee Altmayer, Frank Blakely, Mark Boles, Dave Decot, John DeGood, Joe Malin, Dan Miller, Alvina Nishimoto, Jim Nissen, Barbara Scott, Glen Shirey, Chris Smith, Yvonne Temple, Tim Tillson, Erik Torres, and Jack Walicki. We also thank the R&D labs that contributed their inspections-use data so we could develop the baseline inspections penetration, use, and extent-of-adoption metrics. Finally, we thank the 20 reviewers of this article for their comments, corrections, and enthusiasm, especially Jan Grady, Debbie Caswell, Cate Meyer, and Brian Sakai.

REFERENCES

1. E. Rogers, *Diffusion of Innovations, 3rd ed.*, Macmillan, New York, 1983.
2. M. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, No. 3, 1976, pp. 182-210.
3. T. Gilb, *Principles of Software Engineering Management*, Addison-Wesley, Reading, Mass., 1988.
4. R. Grady, *Practical Software Metrics for Project Management and Process Improvement*, Prentice-Hall, Englewood Cliffs, N.J., 1992.
5. C. Prahalad and G. Hamel, "The Core Competence of the Corporation," *Harvard Business Rev.*, May-June 1990, pp. 79-91.
6. S. Redwine and W. Riddle, "Software Technology Maturation," *IEEE Conf. Software Eng.*, Aug. 1985, pp. 189-200.



Robert B. Grady is the software-metrics program manager for Hewlett-Packard's Corporate Engineering Software Initiatives. He has written and coauthored numerous papers and articles on software subjects, including the books *Software Metrics: Establishing a Company-Wide Program* (Prentice-Hall, 1987) and *Practical Software Metrics for Project Management and Process Improvement* (Prentice-Hall, 1992).

Grady received a BS in electrical engineering from Massachusetts Institute of Technology and an MS in electrical engineering from Stanford University. He is a member of the IEEE Computer Society.

Tom Van Slack is inspections program manager for HP's Corporate Engineer-ing Software Initiative. He has led the inspections program since it was established in 1990.

Van Slack received a BSc in computer science from California Polytech-nic State University at San Luis Obispo. He is a founding member and cochair of the board of directors for the Software Inspection and Review Organization.

Address questions about this article to Grady or Van Slack at Hewlett-Packard, Bldg. 5M, 1501 Page Mill Rd, Palo Alto, CA 94304.

Was Your Last Software Project Late?

If your last software project was late, you need **Costar**, a software cost estimation tool that will help you plan your ***next*** project.

COCOMO is used by thousands of software managers to estimate the cost, staffing levels, and schedule required to complete a project — it's reliable, repeatable, and accurate.

Estimates are based on 15 factors that determine the effort required for a project, including:
- The Capability and Experience of your Programmers & Analysts
- The Complexity of your product
- The Required Reliability of your product

Costar is a complete implementation of COCOMO, so it calculates estimates for all phases of your project, from Requirements through Coding, Integration, and Maintenance.

Costar supports:
- Incremental Development COCOMO
- Detailed COCOMO
- Calibration to your environment
- Ada COCOMO
- Traditional COCOMO
- Function Points

Costar runs on MS-DOS. Site & Corporate Licenses available.

Softstar Systems
(603) 672-0987
28 Ponemah Road
Amherst, NH 03031

Call for a free demo disk.

SOFTSTAR