CENTER FOR QUALITY OF MANAGEMENT JOURNAL

REPRINT NUMBER:
RP05400

Volume 4, Number 1 **Winter 1995**

ISSN: 1072-5296

The Systematic Development of Skill as a Basis for Competitive Product Development

David Walden

This paper is the extended text of a presentation that was given on October 4, 1994, to the R&D organization at Bose Corporation as part of one of Bose's "quality days." The presentation was intended to provide informal explanation and motivation regarding TQM and its relevance to R&D, not as a "cookbook" of R&D methods.

Introduction

Thank you for having me here today. It's a privilege to be meeting with R&D people with as great a record of success as you have.

I'm not here to tell you what you should be doing or how—when it comes to quality, I feel more like a student than a teacher. However, I have practiced engineering and engineering management extensively, and I have extensively studied and attempted to apply quality methods to engineering. Today I'll give you my observations on the practice of quality methods and their relevance to engineering, as well as my thoughts on how one might apply them effectively.

These thoughts are in a state of development in my mind. In fact, I welcomed the invitation to participate today because it gave me a deadline to get these thoughts on paper and thus make them more concrete. I'll welcome any feedback any of you have in response to these thoughts.

I believe you are all involved in R&D. I will sometimes use the word "engineering" to be synonymous with R or D, hardware or software, and anything else any of us do in an R&D organization in a high-tech industry.

My presentation today will cover three topics. First, in practically every field, competitive performance depends on individual mastery (which is not the same as being world champion). Second, TQM is a set of methods and practices intended to allow companies to achieve business mastery of a sufficient level to compete successfully in a rapidly changing world. Third, attaining and maintaining R&D mastery in a rapidly changing world is a particularly difficult problem that will require great intellect and effort from the R&D organization in a form, I think, that is not dissimilar in many instances from the acquisition of mastery in many other fields.

My emphasis today will be on mastery of individual skills. In fields that require team effort (such as R&D), mastery of coordinating the roles of the various members of the team is also necessary, in addition to appropriate levels of individual mastery by the team members. Much that applies to individual mastery also applies to team mastery.

Section 1: Mastery Comes from Self-improvement

I'll address these points in turn, starting with the idea that competitive performance depends on individual mastery.

Performance at superior levels in all fields depends upon individual mastery of the topic area. This is certainly true in such areas as golf, tennis and other sports, bridge, chess and other games, ballet, instrumental music, juggling and other performing arts, physics, math, sailboat racing, airplane flying, war fighting, . . ., and I believe it is also true in software and hardware development, which are the basis for developing the kinds of products I am familiar with.

We can turn this idea around. We consider a person to be an expert in an activity—to have mastered an activity—when the person can reliably outperform better than all but a couple of percent of the population engaged in the activity.[1]

Superior performance has little to do with luck. Over the long haul, and probably in the short run as well, the non-master is simply no match for a person who is a master; neither is the somewhat less skilled person a match for the somewhat more skilled person over the long run.

The individual mastery that is the key to competitive performance is inevitably an acquired capability; for all practical purposes, even the person with the most natural talent is not born with mastery but rather has to acquire it through extended effort.

Again, I am not talking about what is required to become world champion in a field, which some will argue requires natural talent.[2] Rather, I am talking about a level that is the equivalent of being able to shoot par on a golf course, to play even with a tennis teaching pro at a local tennis club, or to learn a new dance and perform it competently with a ballet company. Many people without what people sometimes call natural talent achieve this level of skill that I am calling mastery.

---

[1] Ericsson and Charness, 1994, note that a plausible definition of an expert is someone who performs at a level more than two standard deviations better than the mean of the population of participants in a field of endeavor (i.e., top ≈ 2.25%).

[2] World champions (grand masters) play at a level significantly above experts (masters), i.e., a few more standard deviations above the mean.

As I look at various fields that people master, I notice several characteristics of how people gain mastery.

1. There is typically a prior tradition of skill that it is presumed one must learn to become a master.
2. Serious students typically study with masters, or at least study the methods of masters, to learn the prior tradition and best current practice. Accomplished musicians and dancers tell you who their teachers were and their teachers' teachers. Accomplished baseball and basketball players can explain whose swing or shot they copied. Dennis Connor says to copy the methods of the fastest sailboat racers in your class and get going as fast as they can go before trying to develop your own improved methods.
3. There is typically a relatively common language and notation for the field, which allows practitioners to discuss their topic in detail and to communicate their thoughts to others.
4. Activities are separated into performance and practice, and these two activities have different purposes. Performance is directed toward accomplishing a job or beating a competitor. Practice is directed toward learning new skills, honing old skills, or correcting breaks in form.[3] Repetition through performance alone is not an effective improvement method, and in fact can solidify poor form.
5. After performance and frequently also after practice, there is immediate review for the purpose of understanding what worked and, particularly, what didn't work and why not. This happens after bridge tournaments, chess games, rounds of golf, musical rehearsals, and war games.
6. A major purpose of all this practice and evaluation and modification of performance is to make the skills reliably repeatable—to turn them into a process that can be successfully duplicated every time (and thus leave available some physical or mental capacity to deal with exceptional situations or to observe the big picture). Larry Bird shot hundreds of practice foul shots the same way each day, enabling him to make his shot even after being severely jolted and possibly a little dazed or a little injured from a flagrant foul.
7. In fact, there is usually an emphasis on a few fundamentals, mastery of which can move one past a large percentage of the others involved in this activity and which are a necessary base for reaching the highest levels of skill. Tennis teacher Vic Braden says to learn to reliably hit the "same old boring winner" deep down the center and you'll "be famous by Friday." Golfer Jack Nicklaus says that he doesn't believe in systems—he believes in fundamentals.
8. There are typically a variety of different methods or schools for teaching the same fundamental skills, or emphasizing different aspects of them. Note, for instance, the Horton, Graham, and Cunningham techniques in modern dance.
9. Even at high levels of mastery (and certainly at lesser levels of skill), coaches are available who objectively watch for breaks in form or weak skills. If such coaching is not available, practitioners often develop methods of self-coaching whereby they look objectively at their own performance.[4]
10. In the course of developing high levels of skill, students frequently work their way through a variety of positions on a team, thus learning the skill from all perspectives, or they make significant changes in their game strategy as their skill improves. In sailboat racing, one may start by repacking sails in the bowels of the boat, get promoted to trimming a sail, grow further to calling tactics, and finally end up skippering a boat.
11. Most fields have essentially conservative traditions, teaching what has worked before and changing only

David Walden has been doing and managing research and development for 30 years.

[3] Ericsson and Charness, 1994 (see References and Bibliography), say that practice is for "restructuring of performance and acquisition of new methods and skills."

[4] For example, they learn to keep their own training plan and practice log, they learn to follow a (perhaps mental) check list that reminds them to maintain good practices in the heat of competition, they learn to turn off their competitive juices during post-performance review and practice being brutally objective, analytical, and systematic in finding and eliminating their weaknesses, and they learn to discipline themselves to work hardest on the parts of their game they like least which are probably therefore the weakest parts.

gradually over time.

12 In many fields there is a quantitative result that allows one to distinguish among levels of skill and between master and non-master, e.g., Martina Navratilova's win/loss record against Pam Shriver in tennis, or a 2200 rating vs. a 1800 rating in chess. In cases where there is no quantitative result, there is often an obvious qualitative result, such as the difference in grace of movement we see between a professional ballet dancer and a less skilled amateur.

13. Furthermore, in fields that require team effort, individual team members, in addition to being skilled in their roles, must be assigned to roles that complement each other. People must learn and take responsibility for different functional roles, some of which are less desirable than others. Some people on a team will have already achieved their full potential, while others will still be developing their skill, and still others may be slightly past their prime. Furthermore, where possible people should develop secondary skills that permit them to back each other up. Role playing often requires subordination of self to the good of the team. Only one person can lead a team. There may be another person on a team who is better at my favorite position than I am, requiring that I play a position that I like less well. I may have to defer to less skilled performers from time to time to give a more junior person needed experience. In general, I must play the role that the team needs most at the time and subordinate my own preferences to the good of the team. It frequently takes a very talented leader to organize and motivate such role playing; having a person who understands what needs to be done and can get others to do these things is a key to successful team performance.

We all know how difficult it is to do something that requires repeated or continuing effort, e.g., to learn to play the piano, to actually improve our golf game, or to lose weight. We talk about it, and maybe we try a new beginning now and then. But mostly we fail to reach and maintain any significant plateau. Even if we put in a year or two at the beginning to reach a level where we can feel we know what we're doing and can have some fun, after that we mostly just play at it and don't achieve real mastery.

Thus, most people working in an area are performing at a level lower than the ultimate level they could reach if they applied themselves more diligently.

For instance, Figure 1 (facing page) suggests[5] the distribution of golf skill among all golfers. Across the bottom we have the handicap or average number of shots a golfer shoots above par. As you can see, only about one to two percent of golfers actually shot par on a routine basis, i.e., qualify as fully expert golfers. A few additional percent of golfers come within five or ten shots of par on a regular basis. The vast number of golfers shoot between eleven and forty shots over par on the average. And then there are a handful of golfers who shoot even worse than this. From par to the right is the realm of the non-touring professional and non-top amateur golfers. On the left of the figure is the realm of the touring pros and the very best amateurs. These players regularly shoot below par and have negative handicaps on normal courses. Thus, Figure 1 indicates that in a field such as golf there is much room for improvement for most golfers.

Even though for most people golf is not a full-time occupation and business is a full-time activity, I believe that in business most people, e.g., in R&D, are performing at a level at least a little lower than they could if they worked harder or smarter at self-improvement. For instance, because we in R&D all work full time at it, mostly went to good schools, and were selected by companies known for choosing only excellent talent, let's assume that we have no one with the equivalent of a handicap over 25, that is, we are all at least in the better half of the population. Most of us still have a lot of room for improvement. How many of us in R&D would honestly say there is nothing we can do to improve our skills and performance?

In golf, and most other fields of endeavor, so-called "natural talent" is not a limitation for most people. Most people with enough practice, coaching, and discipline could raise their golf scores to within, let's say, ten shots of par on the average. It may be that a certain degree of natural talent is necessary to reach the realm of the

[5] I got this data orally from an expert golfer, but not from an authenticated source. Still, it's not a surprising distribution, and intuition would probably suggest to most of us a similar distribution for performers in many other fields.

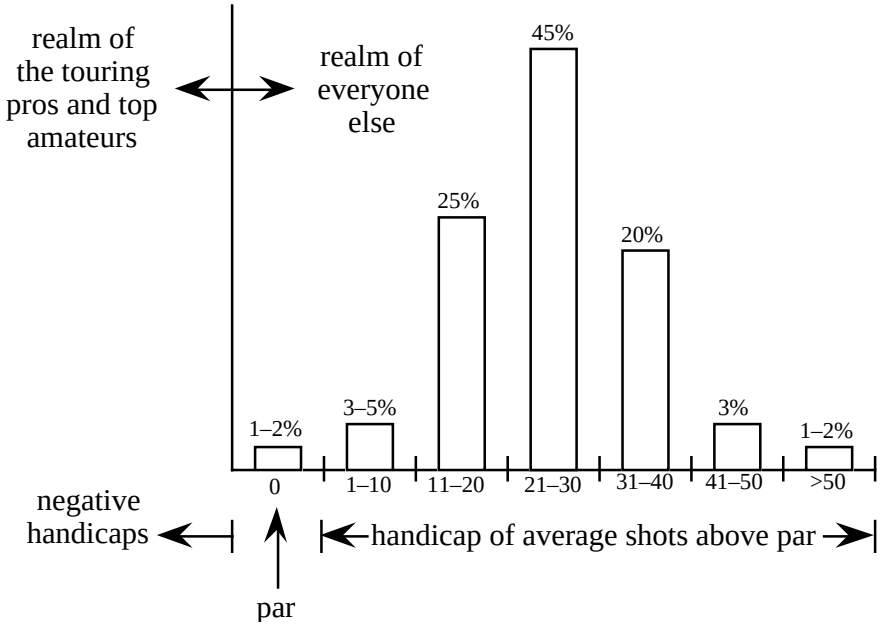

Figure 1

touring pros and top amateurs, but that is not required for success by most people.

In fact, in golf as in most fields of endeavor, what I like to think of as the "process talents"—the talent of motivation, the talent of diligent self-application, the talent for finding the right instructor at the right time—these talents are typically more important for all practical purposes than what many people call natural talent. Not having sufficient natural talent is inevitably an excuse for not bothering to learn or not having the time to learn the form that works for the masters.

In *The Boys of Summer*,[6] the author, Roger Kahn, tells the story of meeting with retired Brooklyn Dodger ball player George Shuba and complimenting him on the "natural swing" he was known for during his playing days. Shuba replied that Kahn could have had a natural swing too, and he took Kahn to the basement of the house in which he'd lived while playing baseball. Hanging from the ceiling was a knot on the end of a rope on which Shuba practiced his swing by hitting the knot. Around the room were little pieces of paper with tally counts, 𝍸, where each tally stood for 60 swings, 600 swings a night, 4200 swings a week, 45 to 50 thousand swings every winter.

Roger Tory Peterson, who invented the modern bird watching guide and in effect created bird watching as a spectator sport, and who has drawn all his own bird portraits for 60 years, still says that he is mastering his craft of painting; "If you want to be the best, you have to practice a lot."[7]

Section 2: TQM is for Improving a Company's Methods to Meet Changing Circumstances

Let's now turn to the topic of TQM and its purpose.

TQM is a set of methods and practices, developed primarily by practitioners in industry for the purpose of achieving business success in a rapidly changing world. In other words, TQM has the goal of providing companies with the skills of business mastery in our modern world. In this way TQM parallels the effective methods that have been developed to achieve individual mastery in various fields. TQM is not an abstract philosophy or theory. It is completely pragmatic, and it evolves as necessary to accommodate the changing business environment.[8]

[6] Kahn, 1972.
[7] Article on Peterson by Scott Allen, *Boston Globe*, October 10, 1994, pp. 25, 28, and 29.
[8] Many of the ideas in this section are taken or adapted from Shoji Shiba et al., 1993.

The methods of TQM seem new and unusual to many of us because they are so different from the methods that most of us have used in business for years. The methods that most of us have grown up with and used for a significant portion of our working lives, e.g., five to forty years, were the methods for business mastery in the industrial era. These methods included mass production, a hierarchical organization, management by objectives, economic order quantities, cost accounting, and mass marketing.

The industrial era started in the 1700s. It really reached its peak of performance in the period starting just after World War II and extending through, say, the mid-1980s.

This was an era when the focus was on mechanical reproducibility; in the world at large communications were generally weak, at least compared with the level of communications we see today. Because of the emphasis on mechanical reproducibility and because of weak communications, the industrial era was one of relative stability or slow change compared to what we see today. The product life for most products was at least five years (and perhaps ten or twenty) without substantial change.

Sometime in the last few decades, certainly since the invention of the transistor, we have begun to cross from the industrial era into the information era. In fact, early in this decade the average annual investment in American industry in information processing assets surpassed, for the first time, the average investment in manufacturing assets (just as the average yearly expense of manufacturing assets surpassed the expenditure of agricultural assets during the previous transition from the agricultural era to the industrial era).

A primary characteristic of the information era is that news travels fast. There are no secrets any more about products or processes. A company no longer has its own nation or geographic region to itself and can no longer dominate its own region, ignoring perhaps a superior manufacturer in another country.

Another characteristic of the information era is that information assets are becoming more valuable than physical assets. For instance, we pay a few hundred dollars once every decade to buy a new TV set, yet we pay a few hundred dollars yearly for cable access through that TV set. Information products can change much more readily in many cases than physical products.

As we get deeper and deeper into the information era, the world is changing more and more rapidly. New user needs are discovered, new products and services are created to satisfy those needs, and new processes are created to make the new products and services or to provide the old products and services more efficiently.

Of course, it is not clear which comes first, the new user need or the new product or service, but it hardly matters. In either case things are changing very rapidly. The average life cycle of a product has decreased from five or ten years down to one year, and for many products it may now be in the range of half a year or less. For instance, in my field, the work station vendors such as SUN or the desk-top computing vendors such as Compaq or Dell bring out several new products every year, and companies that we think of as some of the best in the field, such as Hewlett-Packard, have an explicit corporate practice of cannibalizing their own product lines before other companies cannibalize them. I once heard Bill Gates of Microsoft say that the key measurement of how effectively Microsoft is operating is the rate of new product introduction, because all their existing products are already on their way to obsolescence.[9]

This change is hard to keep up with. We have a hard time figuring out what to do next, or we can't afford the effort to keep up, or our current game is replaced by a new game—the minicomputers game by the workstations game, for example.

Thus, companies everywhere are facing crisis. For some of us, the crisis is current. In all but a few of these cases, the companies recognize this crisis and are trying to grapple with it as best they can. In other cases, the crises may be imminent, and we may see this and may be working on it, or we may still be denying that a crisis is nearly upon us. In still other cases, the crisis is latent. Compared to our competitors, we may think we are doing well. We're doing fine in terms of revenue growth, market share, and profits. Therefore, we may think we have no problems.

However, in every case, someone or some other company is probably targeting our company at this time. Certainly if we are already facing crisis others are targeting us. They are going to our customers, pointing out our weaknesses, and saying it is dangerous to do business with us. Thus, we not only have the problem of

[9] According to the *New York Times* ("Why Seiko Has 3,000 Watch Styles," October 9, 1994, Section 3, p. 9), Panasonic's "consumer electronic products are now replaced with new models on a 90-day cycle, with older products going to discounters."

overcoming our own crisis but must also deal with the bad PR we are getting from others in our industry or the press, looking to be the first to announce our demise.

If the crisis is imminent, it is also almost certain that at least some of our competitors have noticed our imminent problems and are doing whatever they can to take advantage of our situation and our weaknesses. For instance, they may lower their prices just at the time we ourselves can least afford a price decrease that would result in further pressure on our profits, or they may bring out a new competitive offering just as we're gasping for breath from just finishing development to match their last new competitive offering.

Even if things apparently are going very well, there is probably someone targeting us. It may be a company with more resources than we have, which has decided that we are in a pretty good business and therefore they should make money from it too.

It may be a company that is working harder than we are. We may even see them working harder, but we may dismiss their behavior as socially inappropriate or perhaps even crazy. To use two sports analogies, remember the situation when the American wrestler Dan Gable practiced harder than any wrestler had probably ever practiced before, and his competitors called him crazy. Also, in sailboat racing, the great sailor Tom Blackaller, a world champion, publicly stated that Dennis Connor was ruining the sport because of the meticulous attention to detail and obsessive work ethic Connor used to beat Blackaller.

A competing company may have found an improved process which allows it to introduce a product, or distribute it, or manufacture it, or support it better or more economically than we can, and they are preparing to attack us using this new competitive advantage.

Or another company may have noticed a technical or economic or market discontinuity before we did. This is partly what happened with Toyota when they noticed that there was a demand for economical small cars of high quality and long life, while the American automobile companies assumed they were still in the era of gas-guzzling monsters that their owners replaced every three years.

Some of you may have had an introduction to Total Quality Management or have studied TQM quite deeply. As part of this experience you may have been introduced to the so-called "Four Fitnesses." The Four Fitnesses describe four eras in quality since the beginning of the industrial revolution, or at least since the beginning of this century, and call our attention to the fact that what users require does not stand still.

Fitness to Standard is illustrated by Henry Ford's statement that customers could have their cars any color they wanted as long as it was black. Fitness to Standard is the idea that we in the company make the product the way we think it should be made, and it is the customers' job to buy it that way.

Fitness to Use expresses the idea the customers also want the product to be usable by them *for their purposes.*

Fitness to Cost expresses the idea that customers not only want improved performance and improved delivery; they also want lower cost to them, i.e., the price.

Finally, Fitness to Latent Requirement makes the point that customers in many cases don't yet know what they will want next, but we had better be working on anticipating it or someone else will anticipate it first and beat us to the market.[10]

Fitness to Cost is a particularly important concept for R&D organizations to get acquainted with, in my view. Although we are not alone, I think we in R&D hold most dearly the view that if we can make a better product—that is, a more powerful product or a product with more features—we should be able to charge a higher price for it. We use this idea to convince ourselves (and sometimes our managers and our marketing and sales people) that we are doing the right thing when we keep adding functionality and power to the product, although it expands the development time and cost.

Our idea that people will pay more for a better product is based on the assumption that **cost + profit => price**—that we get to choose the cost to go along with this outstanding product performance, that we then add a fair profit, and this gives us a price the customer should be willing to pay. However, today's reality is that the market and the competition set the price, and they may set the price below what it costs us to make our superior product. Therefore, we must make our product at a cost such that when it is subtracted from the market price we still get a profit. The market controls the price, we control the cost,

[10] The idea of "mass customization" that is also part of our transition into the information era may be covered by the idea of Fitness to Latent Requirement, or maybe it's part of the development of a new fitness. Mass customization is the idea that we provide a different product or service for every customer rather than the same product or service for a mass market. This is consistent with the idea of relationship marketing, which says that our job is not to win market share but to win breadth of business from each customer.

and thus we get a profit, or **price – cost => profit**. However, we can't simply cut out functionality because we lowered our cost. We must provide the same functionality chosen by our competition, or more than our competition does at the same price—even though they may be larger and have greater economies of scale, or may be subsidizing their product that competes with our product through sales of another product line (which we perhaps do not have), or may be using any of a number of other mechanisms that we consider unfair.

Thus, TQM is a set of evolving management methods designed to help companies succeed in a rapidly changing world.

As illustrated in the Figure 2 (below), the methods of mass production were designed for managing with maximum efficiency in a slowly changing world. The world on the right of this little diagram stands still, and the company is basically a massive optimization system with all of its efforts focused on efficiently producing a product that addresses this static world.[11]

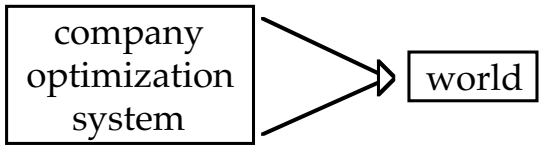

Figure 2

In the world that TQM envisions, the world is moving, as shown on the right of the Figure 3 (above, right). Therefore, rather than the company's management system being an optimization system, it must be a set of learning techniques that obtain feedback by observing the changes in the world and changing what the company is doing to produce a product narrowly targeted at where the world is today.[12] If this is not a familiar concept to you, I strongly urge you to read the book *The Machine that Changed the World*, by Womack and his colleagues at MIT.[13] I think it is a tremendously important book. Certainly every R&D person I know who has read the book has come away sensitized, and some have been profoundly moved by it.

The Center for Quality Management (CQM), of which your company is a founding member, has a way of teaching TQM that divides the methods of succeeding in a rapidly changing world into four categories. The CQM calls these

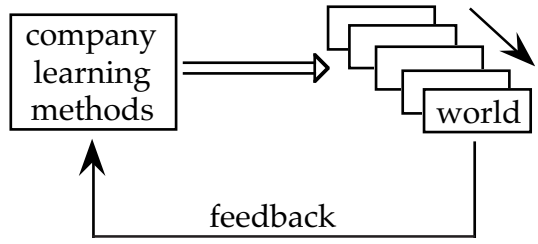

Figure 3

the Four Revolutions in Management.

The first revolution is Customer Focus. This notion is in contrast to the idea (which should be familiar to all of us who come from the R&D world) that we in the company know what is best for the customers. There are many reasons for customer focus. For instance, if the world is changing rapidly, then we had better keep our eyes on that world, or else we will not know how it has changed and therefore how we have to react. Another reason for customer focus is that all companies (in this rapidly changing world where we can no longer do the same thing year after year) have insufficient resources to do everything they need to do. Therefore, we had better make sure we know what is happening in the customer world so we can, at the minimum, apply our resources to those areas which are necessary to keep our existing customers and get new ones (even if it means we have to forego building an improved cost accounting system).

Continuous Improvement, the second revolution, is the idea that we have got to keep changing and improving our methods. This is in contrast to the old idea that "if it ain't broke don't fix it." As with customer focus, there are a number of reasons for continuous improvement—the competition may be getting stronger, our products or methods may be getting obsolete, it may not be clear exactly what the final target is and therefore we are better off doing a series of incremental improvements and getting customer feedback after each, etc.

The third revolution is the idea of Total Participation. This is in contrast to the old idea that some people were responsible for developing the methods a company used, and everybody else was responsible for just doing their job and using the methods assigned to them. Total participation is necessary in a rapidly changing world in order to react fast enough. If you've got people

[11] TQM calls the idea expressed in this figure "Product Out."
[12] TQM calls the idea illustrated in this figure "Market In."
[13] Womack, 1990.

back in a lab somewhere who are the only people responsible for changing your methods, and you've told the person who is dealing with the customer every day that their job is to follow company policy, then you are not going to have a system that quickly delivers the news that the customers are now asking for something new, and the current procedure is no longer working. Furthermore, in today's competitive environment, we can no longer afford to squander the asset that is the improvement capabilities of the majority of our employees, particularly since these are the employees who are actually doing the job, and therefore have the best data on how the job might be improved, or whether the job correctly addresses its purpose, or what the customers are saying.

Finally, the fourth revolution is Societal Networking. This is in contrast to the idea that we've got to go it alone because we've got to protect our competitive position. The fact is that things are changing so fast that if we try to go it alone we will find ourselves quickly obsolete. None of us has the resources to develop the new methods as fast as we need to. We are all forced to ally ourselves with others, either formally or informally, and in so doing we will share some of our methods. However, the wonderful aspect of this societal networking is that we typically get much more benefit back from *all* the people we share with than any one of us gives. (This, incidentally, is analogous to the tradition of scientific publication in the western world that has proved so effective over the last few centuries.)

The theme that I believe runs most strongly through TQM and through the Four Revolutions in Management is systematic development of skill. Systematic development of skill is in contrast to hoping for or waiting for so-called "natural talent" to appear. Systematic development of skill says that we—as individuals, in our teams, in our companies, perhaps in our regional industries—can and should take responsibility for our own success by making ourselves more competitive. Systematic development of skill says that we're not going to be resigned to the fact that somebody else has a lesser handicap than we have and we are doomed to always follow them. Systematic development of skill says that with appropriate motivation and discipline, we can improve ourselves so that we eliminate some of our handicap and become more competitive.

I believe that in virtually every field of endeavor the factor that most directly separates winners from non-winners is the relative level of mastery of the field that the competitors have.

I also firmly believe that a higher level of mastery can inevitably be acquired through self-improvement work. Each one of us, however good we are, with the possible exception of the few people in our companies who may be already operating at the world-champion level, has the capacity to improve, perhaps not to the world-championship level but by a significant percentage. If each of us in our company improved by only five or ten percent, I suspect that would be a tremendous change in competitive advantage for our company. And I think many of us have the capacity to improve by 50% or more. In my field of software development, it is generally believed that the productivity of individuals varies routinely by factors of 10, and that factors of 100 or more are not so unusual.

The key to self-improvement, however, is that you have to actually change what you are doing. It is easy to want to change; it is difficult to actually change. Although Watts Humphrey, in his book on software process, quotes it, I believe it was in one of Rita Mae Brown's novels that I first read this definition of insanity: "insanity is doing the same thing over and over again and expecting something to change."

Thus, TQM has us decide what level of performance we want to attain (typically a manageable step rather than a pipe dream). Having decided what level of performance we want to attain, we do analysis to decide what needs to be improved—what actually is blocking the level of performance we want. Then we decide how to go about improving that aspect—how should performance actually be modified or what new skill or method should be learned.

Then TQM has us do the skill-building work and try the new method, and monitor if we are actually doing it. Then we test to see if things actually got better. If things got better we include the new method or exercise in future efforts. If things didn't get better we figure out why what we tried didn't work. [Was it that we mistook the level of performance that was necessary, or that we mistook the area that needed to be improved, or did we find the correct target and the correct area of improvement but simply fail to do the necessary skill building? In my experience, the last is most often the case. We make a perfectly good plan for improving what needs to be improved and then never carry out the work necessary to actually make the improvement.] And finally, we restart this cycle to do it all over again, to continue to ratchet up our performance.

PDCA Cycle

Plan	• Decide what level of performance you want to attain.
	• Decide what actually needs to be improved.
	• Decide how actually to go about improving it.
Do	• Actually do the skill-building work.
Check	• Actually test if things got better.
Act	• If yes, actually include this method as appropriate in future efforts.
	• If no, actually figure out what was wrong (e.g., performance target area of improvement, improvement method, skill-building effort).
	• Actually restart cycle.

Figure 4

7 Steps

Step 1	• Decide what level of performance you want to attain.
Step 2	• Decide what actually needs to be improved.
Step 3	• Decide how actually to go about improving it.
Step 4	• Actually do the skill-building work.
Step 5	• Actually test if things got better.
Step 6	• If yes, actually include this method as appropriate in future efforts.
	• If no, actually figure out what was wrong (e.g., performance target area of improvement, improvement method, skill-building effort).
Step 7	• Actually restart cycle.

Figure 5

In TQM the steps enumerated above are known as the PDCA cycle, standing for, Plan, Do, Check and Act. (See Figure 4, above.) The first three steps are the Plan part. The next step is the Do part. The next step is the Check part. And the last two or three steps is the Act part, deciding what should be done next either to standardize on the new method or to try to find a different method, after which the cycle is restarted.

For those of you who are familiar with the 7 Steps, the cycle I have just described also is essentially the 7 Steps (see Figure 5, above). The 7 Steps is one form of PDCA that is used for reactive problem solving—improving a process that we want to keep but want to make work better. Reactive problem solving typically is not used to develop new processes, but it is very useful in improving many types of existing processes.

The essence of "PDCA," "the 7 Steps," and other such TQM code words is that they are devices to teach those who don't know (and remind those who should know) about good scientific problem solving.

We could debate the philosophy of the scientific method all day (and philosophers have done it for a very long time); however, I think that we can agree that for practical purposes the scientific method has two aspects.

First, it alternates between theory and practice, between thought and data, between abstract ideas and empirical work. Theories are subject to verification. Using the scientific method in the case of self-improvement activity, we sense a problem, so we gather data. We then analyze what we see and hypothesize an improvement, and then we test whether the fix works. We then standardize on the new improvement method, and then check the data to be sure we stay within spec. That is the scien-

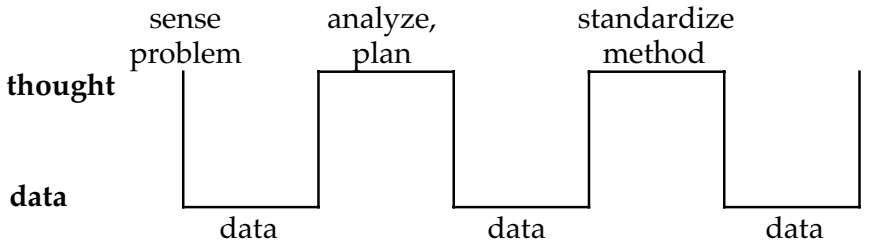

Figure 6

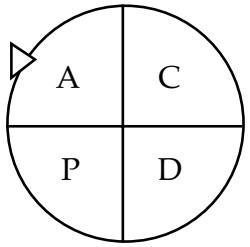

Figure 7

tific method—to alternate between theory and practice (Fig. 6, above). If a theory cannot be subjected to testing, it's not science.

Second, the scientific method is a never-ending process. We build newer theories on previously proven theories. That is why TQM shows PDCA as a cycle (Fig. 7, above). The scientific method also recognizes that things will change—that we proved one theory based on theories that we have proven before, but that as we dig deeper we may have to go back and re-explain some of the phenomena that we previously thought we had explained completely.

In many companies an all too typical improvement approach has been the one shown in Figure 8, below. First, we realize we have a problem. Then we dither interminably, with enormous intramural arguments ("Is it this?," "Is it that?," "Is it the other thing?," "Will this affect my job?," "This is my territory"). Finally, through a series of compromises perhaps, we announce a new policy. Then mostly everybody ignores it. No real data was ever taken; there is no iteration. We work entirely at the level of theory or opinion or persuasion or politics. TQM says between each one of these thought steps we have to get data and check whether we have really understood the problem. Having made a plan about what we've got to correct, we need to get data to see both if we are actually correcting anything and if we are actually carrying out the correction process. Once again, the most common failure to improve is not that we haven't found a reasonable improvement method; rather it is that we fail to carry out our improvement plan.

Rapid iteration is particularly important in a rapidly changing world because the targets keep changing. The *results* targets change as what we need to do to satisfy customers, or what competitors are doing, changes. The *methods* (process)

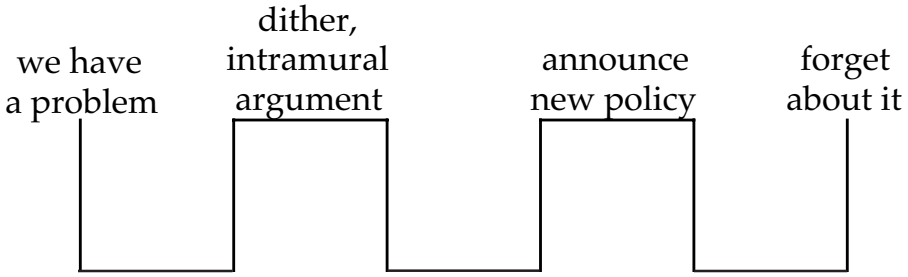

Figure 8

targets change as we have to achieve new results targets. The faster the world changes, the shorter the iteration cycles must be for us to keep our sights trained on the moving target. Also, in all fields of endeavor, skills erode without practice and feedback. With practice alone and no feedback, over time skills will drift; we'll get little hitches in our form. Sometimes these may be improvements; most of the time, I'm afraid, they lower our level of performance. Also, we change. In physical and mental areas people grow older and lose some capability, and they have to develop new methods to compensate so that they can still continue to perform at a high level. Companies change as well, as they go from small start-ups to large companies that have many millions or hundreds of millions of dollars in revenue. The methods that work for a tight little entrepreneurial group no longer work as the company grows large, and they must be changed. Finally, targets are often difficult to see clearly, and we need to iterate rapidly and look often at the apparent target to discover its actual location.

Alternation between theory and practice is also important. Without theory, practice does not accomplish anything specific. Without empirical practice we don't know if anything is actually getting better, or even changing in any way. There is a saying that "if you can't measure it, you can't improve it." This expresses the idea that unless we can tell whether something is changing and, if it's changing, whether it is getting better, we can't get any feedback to help us solidify our skills or adjust the skills we are trying to learn. On the other hand, simply measuring it is not sufficient; in addition to measuring something, we also actually have to change it, and this is one of the important reasons for monitoring the process variables as well as the results variables.

In summary, TQM methods were developed as a result of improvement ideas tested in practice. Things that may look odd (e.g., KJ color and grouping guidelines) are based on vast amount of successful experience (with the benefits of a conventional notation that everyone recognizes, or with bottom-up sorting). Ideas that strain one's credibility—one's mental "muscle memory"—are not so different from the thoroughly proven tennis grip used by all winning players that strains one's physical muscle memory and makes one say that the grip used by successful players everywhere is not appropriate for me.

TQM methods are subject to change, to address special local circumstances or to evolve with global increases in understanding. But these changes must themselves be demonstrated empirically to actually work. On the other hand, it's sufficient that the new methods be validated in a few documented cases; it's not necessary for all users to think they must validate the new methods themselves, and trying to do so often causes people to reject the methods prematurely.

Section 3: TQM or Self-improvement in R&D is Significant Intellectual Challenge

So far, I have discussed the idea that performance in all fields depends on individual mastery, with a little mention of the importance of roles in team activities. I have tried to make the point that TQM has the purpose of providing mastery of the individual and business processes that are necessary for competing successfully in a rapidly changing world.

In this final section, I'm going to discuss the fact that achieving competitive R&D performance in a rapidly changing world is a particularly difficult problem. I suspect, as you have probably suggested to your managers, that it is a more difficult problem than people face in manufacturing or administration. The problem in R&D is not so different that something can't be done about it. However, in the R&D world I think that attaining mastery in the face of a rapidly changing world will require all the intellect and effort that we in R&D (who are familiar with the scientific methods of improvement used by TQM) can possibly give to it.

Why is achieving competitive performance in R&D through individual mastery in a rapidly changing world so difficult? I'll suggest some of the reasons.

In the first place, much R&D is terribly complex. As you know, in R&D we build devices which have hundreds of thousands or millions of transistors in them, we build software which has tens or hundreds of thousands or millions of instructions, and we analyze acoustical situations of enormous complexity.

The R&D craft also is changing practically as rapidly as the world around us. The previous generation of workers may not actually be completely capable of teaching or mentoring the next generations of workers in software or hardware design. It is quite possible to be five or ten years out of school, be leading a significant number of engineers for the first time, and find that one's skills are becoming obsolete.

In R&D there can be a terribly long feedback cycle, at least the way we typically set up projects. Projects may go for months or years,

and then at the end of that time we pat ourselves on the back because we have a "lessons learned" review. Then the product goes into the market, and it may still be more years before we discover whether the customers really want to buy the product or not, and before we learn all of the support problems or performance problems the product may have.

As we develop a variety of methods in hardware and software engineering and the other disciplines that we use in R&D, it is difficult in many cases to correlate the use of these methods with the results. Thus, it can be difficult to discover what works and what doesn't.

Furthermore, what is needed most is scientific investigation—an attempt to use new methods and new metrics—yet we live in a corporate context that doesn't particularly understand or value trial and error, effort spent learning rather than working, practice rather than performance, or time spent coaching. Neither the managers nor the individual engineers (because of the way we teach them in school or the way we brainwash them in business) understand the tremendous value in these sorts of activities.

In R&D we have few accepted standards of performance, and some of the ones we have aren't very good. For instance, I can point you to a book[14] on why the measure that is most often used for evaluating programming performance, namely lines of code written, is at best irrelevant and at worst catastrophically misleading. Unlike golf, where there is a quantitative score, or ballet, where you can see if the dancers stumble when they pirouette, in engineering we tend to work in little corners by ourselves, and it is difficult to evaluate who is actually very good and who is only average. I suspect that some people we think of as great R&D experts are really not particularly more skilled than others whom we don't think of as experts.

Finally, we have a poor understanding of the value of finding the best roles for people and motivating them to flourish in those roles.

As I said, achieving or maintaining competitive performance in a rapidly changing world in an R&D organization is going to require great intellectual effort from everybody in R&D. But I think we can't simply say that R&D is too hard to improve, or that it all depends on individual talent, or that there is no way to apply systematic methods to R&D. First, I don't think these things are right; and, second, such an attitude will doom us to falling behind competitively.

I am now going to describe one specific instance where a company understood how to do systematic development of skill in an R&D. That is the case of the NEC Integrated-circuit and Microcomputer Systems (NIMS) division. Then I am going to finish with a number of ideas of my own.

The NIMS division of NEC won the Deming Prize in 1987 for their work in quality improvement. This effort was lead by Kiyoshi Uchimaru, who was a professional engineer for nearly four decades, until he became president of the thousand-person NIMS division. This division started out as a body shop that rented out engineers to other parts of NEC, and it thus had no capability to manage projects or do complete technical designs. Uchimaru decided that if they were going to survive, even within NEC, they had to become self-sufficient. You should read the book *TQM for Technical Groups* by Uchimaru and his co-authors, published in English by Productivity Press in 1993. (Incidentally, Mr. Uchimaru died in the fall of 1993; in my view we have lost one of the important thinkers on R&D management of our generation. I had the honor of writing the foreword to the English edition of his book.)

Uchimaru came by a lot of his ideas by studying professionals he observed in various fields, and I suspect that one reason I have emphasized a number of fields outside of engineering as I have talked about systematic development of skill today (along with being interested in games and sports myself) is the influence that Uchimaru had on me when I met him on two different occasions.

Uchimaru defines a professional as the equivalent of someone who can shoot par in golf—in other words, someone who can go around 18 holes with a score of 72. He makes the point that while most of us in R&D call ourselves professionals, what we really mean by the word "professional" is that we get paid for working full-time in the field. Uchimaru does not define this as professionalism; he defines it as "full-time." Many people in most engineering organizations are less than professionals in Uchimaru's sense.

In Uchimaru's experience "pros" in many different fields have three distinguishing characteristics:

- They have a strong grounding in theory
- Because of their long experience and

14 Jones, 1991.

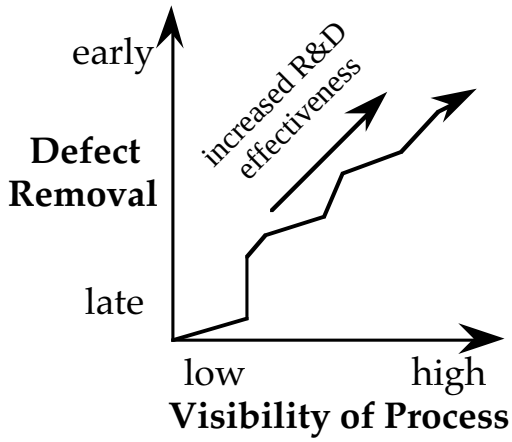

Figure 9

the many PDCA cycles they have experienced, they have many tools available to them and the ability to use the tools together in flexible ways

- They have a strong ability for self-analysis of failure.

This last point—self-analysis of failure—is in some ways the most important characteristic, for it is by self-analysis of failure and concentrating on eliminating weaknesses that over time one brings one's skill to the highest level.

Uchimaru, with his strong engineering background, personally led the NIMS improvement effort. When he started out on this improvement effort he said that he did not understand how to apply the quality methods of manufacturing and administration to R&D, and instead he came up with two basic principles which they used throughout their improvement work.

One of the two fundamental principles was the idea that it is better to have a more visible process; this is shown in Figure 9 (above) on the horizontal axis, where visibility of process goes from low to high. His other basic principle was that it is better to remove the defects early rather than late in the process; this is shown on the vertical axis. Uchimaru's goal was to find a succession of process improvements that would either improve the visibility of the process, remove the defects earlier in the process, or both. Like so many R&D groups, NIMS was at the lower left corner of the graph when it started its process improvements. Its process was invisible. Each engineer had his own process in his head (to the extent they had any process at all). And they sometimes removed the defects at the last possible moment, e.g., after they had shipped the product to the field. However, through a series of improvements they gradually worked their way to the right and upward, until eventually they had a very visible process and the defect removal was done at the earliest possible moment.

There is not time now to go through the whole NIMS story, and you can read the book. However, the series of improvements that they put in place included the following:

- A metrics program so that they could find out what was happening
- Design reviews so that they could catch bugs in the design earlier
- Design plan reviews before a project started so that they could discover that their development plan had flaws before they carried it out
- Quality Function Deployment to get the voice of the customer matched to the implementation of the product
- "Neck engineering," where they made sure they had the means of obtaining fundamental technologies which were not available off the shelf, in time for when they were needed within the project
- Defect prevention, where bugs found in products in the field are traced to their source in the development process, and the process is changed
- A detailed phase review process with explicit models and specifications at the beginning and end of each of the four or five development phases most of us are familiar with, as well as numerous intermediate phases, so that they could catch bugs earlier, make the process more visible, and thus find out what was working and what wasn't, in much greater detail
- Finally, they taught problem prediction to their engineers using a method they called "event management." In this process, every three or four days a master engineer sits with each younger engineer on the project and helps him or her develop engineering skill by learning to recognize design risks and to anticipate them sufficiently to avoid making a mistake.

Thus, in addition to instituting a number of other development techniques, they ended up with a program that works on the systematic de-

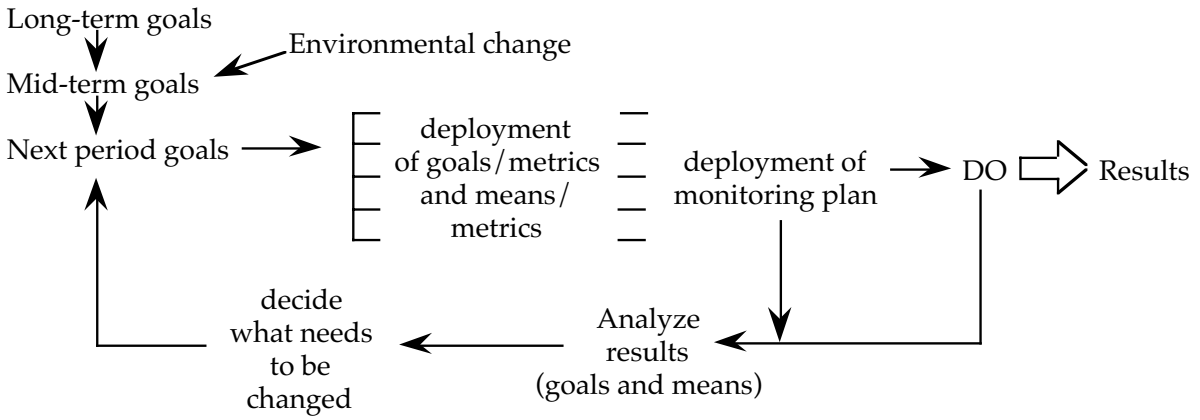

Figure 10

velopment of engineering expertise with the goal, to use the golf analogy, of significantly reducing the handicap of every engineer in the organization and bringing everyone as close as possible to the level of a professional.

The method that NIMS used to run their PDCA cycle was policy deployment, which I understand you are beginning to try here. In the CQM, we also call this Hoshin Management (Fig. 10, above). NIMS used Hoshin Management to run their improvement experiments, to carry out controlled experiments, and to decide what needed to be done in each cycle of the experiment. They put in place the proper targets and appropriate means of reaching the targets, as well as the methods of measuring both the targets and the means to discover if the method succeeded. If it didn't succeed, they had the means to learn whether it wasn't a good method or whether it failed because they didn't actually carry out the means.

As I previously mentioned, NIMS won the Deming Prize in 1987. In addition, their hardware and software productivity levels went up markedly and their defect levels went down markedly.

In the rest of my talk, I am going to share with you a few thoughts that seem particularly important to me. Some of these ideas are well known.[15] Some may not have been proven in R&D practice, as were the methods I described in the NIMS case study, where Uchimaru followed two simple principles—making the process visible and finding the defects early. My selection of methods in this section is based on the simple idea of looking for ways of reproducing in the R&D or engineering setting the characteristics I observe in the way people attain mastery in other fields, such as those I described in the first section of my talk. [Thus, once again, I'm being influenced by Uchimaru and his idea of looking at how people normally learn skills—from a master.] Please give me your thoughts on these ideas.

Take responsibility for own success

My first thought is that we must take responsibility for our own success, and our teams must take responsibility for their success. In companies, many people somehow have the idea that it is the company's responsibility to make them successful. Of course, an enlightened company will do all it can to make a team or an individual employee successful. However, in the end no one cares about the team's success more than the members of the team do themselves, and no one cares more about an individual employee's success than that employee does. I remember interviewing a job applicant who had come from another company for an engineering opening. I asked him to tell me about the technical journals he read, and his response was, "I don't read any because my company doesn't buy them for me." Well, I say it is a fairly unenlightened company that doesn't provide any technical journals for its engineering staff. But it is a far more naive and even stupid person who does not understand that, as his skills erode and he becomes unmarketable, he, not the company, is going to suffer most. In all of the other endeavors I have discussed today, it is the individual practitioner who wants to suc-

[15] I'm mostly not addressing in *this presentation* the larger systems for "managing product development," e.g., phase review systems, concurrent engineering systems, and so on.

ceed, for whatever reason. I think we will make a lot more progress in our R&D organizations if we determine, both individually and as team members, that we are going to do what is necessary to improve our level of skill so we can compete successfully. Certainly, no team can perform outstandingly well if the individual team members haven't developed their individual skills.

Continuous Improvement as a Means of Breakthrough[16]

Next, we must get away from the idea of seeking breakthrough and instead seek continuous improvement. The breakthroughs in which there is a truly sharp break with the past happen so seldom as not to be useful in business. The fact is that practically every breakthrough we talk about is the result of many experiments, many people working over time, and ideas that are in the air; then finally someone takes the step that seems like a very big step, and looks at the problem in a different way, and we call it a breakthrough. All of the great scientists engaged in repeated, detailed investigations and experiments, and probably had many failures before they had their breakthroughs. I believe Edison said "if you want more successes, you have to have more failures," Emerson (?) said "the harder I work, the luckier I get," and Pasteur said "in the natural sciences, success *only* favors the well prepared." Continuous improvement is the means of breakthrough. In fact, the most impressive thing to me about research scientists is their dogged persistence, through routine and through setbacks, until they finally succeed.

Donald Knuth, a computer scientist and mathematician whose research has changed the way people think about the analysis of computer algorithms and who was awarded computing's top prize, the Turing Award, tells a wonderful story.

> I was scared stiff that I wasn't going to make it in mathematics. My advisors in high school told me that I had done well so far, but they didn't think I could carry it on in college. They said college was really tough, and the Dean had told us that one out of three would fail in the first year. . . . At Case, I spent hours and hours studying the mathematics book we used—*Calculus and Analytic Geometry* by Thomas—and I worked every supplementary problem in the book. We were assigned only the even-numbered problems, but I did every single one, together with the extras in the back of the book, because I felt so scared. I thought I should do all of them. I found at first that it was very slow going, and I worked late at night to do it. I think the only reason I did this was because I was worried about passing. But then I found out that after a few months I could do all of the problems in the same amount of time that it took the other kids to do just the odd-numbered ones. I had learned enough about problem solving by that time that I could gain speed, so it turned out to be very lucky that I crashed into it real hard at the beginning.[17]

Curiosity and the Open Mind

Next, I think we cannot underestimate the value of curiosity and the open mind. Learning new methods and improving weakness require an open mind and curiosity. The great scientists and engineers tell us this, and people who are experts in various other fields of endeavor tell us this. However, I have observed that in many of our engineering organizations as in many other endeavors, an open mind and curiosity are not present in great abundance. Many people have this attitude: I do my job and that is all I am responsible for, and then I go home. Others resist the idea that there could be a better method for them than the one they are using now. Once again, we ourselves will benefit most if we improve; therefore, if we have a closed mind and we lack curiosity, of course we are hurting our company, but we are hurting ourselves worse.

I believe we should all read case studies. I'm told that in Japan two thirds of the books published on quality methods are case studies, as opposed to textbooks. Traditionally people have always learned by studying the masters. I think we also need to read about or hear lectures on new methods—to seek them out.

Furthermore, I think it is important, as we study these cases and read these methods, that we accept and understand what we are reading and hearing before we evaluate it. There is a strong tendency for people (all people I think, but particularly people who have gone to engineering school and worked in R&D groups) to evaluate a new idea as they are hearing it, and to try to find the problems with it or to improve it. This means that if the problem is at all complex, there is a good chance they won't have completely understood what's being explained to them. Unless people who are listening give their all to try to bring the idea in, grasp it in its fullness, and look for all its best aspects, there is a good chance they will never actually hear or

[16] See also Walden, 1993.
[17] Albers and Alexanderson, 1985, pp. 181-203.

completely understand the method. There is plenty of time to reject methods after we have learned and understood them. On the other hand, if we reject ideas before we understand them and before we try them, these improvement methods are perhaps lost to us forever.

Go See Customers

Next, I think we need to go see customers. This is the way to learn what game we need to learn to play or to learn which skills need improvement, such as speed, precision, skill in certain technical areas, or that our R&D groups simply isn't producing the products they need.

Sometimes the marketing people don't like us going and seeing customers, and sometimes we don't see why we have to go see customers because, after all, the customers don't understand what is technologically possible. I think there are a couple of very important reasons, however, for seeing the customers.

On the one hand, customers provide tremendous focus and energy for the company. We all have participated in R&D projects where there was lots of intramural squabbling and things didn't seem to get going; and we have also participated in projects where there was a demanding external situation that had to be addressed, and we somehow forgot about our intramural squabbling and our own ideas for how things should be done and pulled together to address the customer's problem. Going to see customers to get that focus and energy from an external source is valuable. We can't afford to lose the time if we are going to remain competitive with those people who have targeted us for destruction. On the other hand, in many cases we find innovations from visiting customers.

Now, as engineers, we often say "customers don't understand the technology, so we have to explain the technology to them and how they can use it." We think we should just design the product using the best technology and then send it out to market, and the customer should buy it. This is all pretty risky because we can spend a lot of time trying to explain technology to customers, who these days are increasingly uninterested in technology and just want *their* problems solved. In fact, most customers I talk to have no interest in the technology. I've been involved in hundreds or perhaps thousands of sale situations in my business career, and virtually never was the technology the deciding issue for the customer. The deciding issue was inevitably something having to do with reducing the level of risk or hassle, or perhaps reducing cost for the customer.

We can also run the risk of spending a long time developing a product, only to have customers reject it because it doesn't really meet *their* needs. von Hippel has noted that in something like 80% of cases, and he has done a number of studies over many years, the customer in fact provides the innovation. Of course they don't provide it as a finished product; that is our job. The customer typically provides the innovation as a "work-around" of some flaw in our existing product, or something that a competitor's product can't do, or something for which there is no product available. Thus, by seeing what the customer is doing we can get a lot of product ideas, and we can bring as much creative energy to bear in figuring out how to use our technology to satisfy the customer's problem as we can in trying to convince the customer to have a problem that uses our technology.

Von Hippel also pointed out that it is important for engineers to go see the customers because, while sales and marketing people are great at hearing what the customer has to say about what they need, they are typically weaker than engineers at seeing what the customer is actually doing. Combining engineers and sales or marketing people on customer visits lets both groups hear what the customers say they need and see what the customers are actually doing. This lets us understand how to interpret correctly what the customer says, and then we can bring that data back in house and use all those creative juices for developing a technical solution that really addresses the customer needs. (Concept Engineering is designed to get this sort of insight from customers.[18])

Gather Qualitative Data

As I mentioned earlier, another problem with systematic development of skill in an R&D organization is the lack of quantitative data. Therefore, I think in many cases we are forced essentially to gather qualitative data as the next best thing to quantitative data, and I call your attention to the two chapters in Donald Knuth's book called, *«Literate Programming»*, listed in the bibliography accompanying this paper. Each chapter is many pages long, and each goes through the logs that Knuth kept for several years of all the bugs he found in his TEX math typesetting program and of how he corrected each bug. He used these logs to teach himself how to program better. How many of us keep of a log of what we do? Knuth keeps an exhaustive log; maybe Knuth is so much

[18] Burchill, 1993.

better than the rest of us in the software world partly because of his discipline in gathering this qualitative data, analyzing it, and therefore improving his skill. Many people in R&D maintain a laboratory notebook. How many of these use it as a source of insight about potential ways to improve personal skill?

Edith Wilson of Hewlett-Packard is another person who has done a systematic study of a large number of cases of development projects, and thus she has some solid qualitative and some quantitative data.[19]

Use Quantitative Data Where Possible

Quantitative data is available that we don't seek out. Those of you who have tried the 7 Steps may have discovered that when they are applied to R&D, frequently there is not as much quantitative data readily available as the 7 Steps process anticipates. However, I call your attention to Grady's two books,[20] in which he discusses putting a metrics program in place. A metrics program could provide sufficient quantitative data. I am sure that those of us in R&D have the mental capacity, if we apply ourselves, to develop metrics that objectively measure what we are doing so we can figure out if we are improving or not and whether we are measuring things that are relevant.

The hardware development world has done a good job in some cases, for instance, quantifying the reduction in debugging time and the reduction in number of respins, through the use of circuit simulators. How many of us in the software world understand the quantitative benefit to debugging time from rigorous use of **lint**, use of **assert** statements, or use of a tool to automatically compile the code for finite state machines?

Some researchers of improvement methods have focused on the quantitative aspect of improvement. Capers Jones, for example, a consultant based in Burlington, Massachusetts, has focused on taking a statistical approach to software development processes. For instance, using statistical data he has collected, Jones has come up with a list of the most serious software risks that hundreds, or perhaps thousands of companies experience on the average. These risks are: inaccurate metrics, inadequate measurement, excessive schedule pressure, management malpractice, inaccurate cost estimating, silver bullet syndrome, creeping user requirements, low quality, low productivity, and canceled projects.[21] Now each of us has seen each of these problems, and we have also seen as many others as Jones has, but how many of us have the data to support the fact that these are the ten most serious software risks?

One of the results of Jones's statistical research is that he has validated the importance of having an active software quality assurance (QA) organization, which is a group focused on bringing more science to software development. By an active quality assurance organization Jones means one that is actively involved with the development project, performing the following functions: moderating design and code inspections, collecting and analyzing defect data, developing and running test scenarios, estimating defect potentials, recommending corrective actions for quality problems, and teaching courses on quality-related topics.[22] Notice that all of these functions have to do with analysis of data, planning how to do experiments, teaching improved quality methods, and so on. Actual testing of software is not part of the QA function, in Jones's view; that is something that specialists in the development (not QA) organization do.

Among the many methods that might be useful in removing software defects, according to Jones's statistics, four are particularly powerful—design inspections, code inspections, quality assurance, and formal testing. The following excerpt from a chart Jones has presented indicates just how powerful these methods can be (Fig. 11, facing page).[23]

If none of these four methods is used, the best efficiency in software defect removal Jones has seen from the development groups he has observed is 50%. The worst efficiency he has seen is 30%, and the median efficiency he sees is 40%. If only the best single method of these four methods is used, the software defect removal percentage jumps to the range of 45% to 68%. If the best two of these methods are used, it jumps to the 70% to 90% range. If the best three of the methods are used, it jumps to the 85% to 99% range. If all four are used, it jumps to the 95% to 99% range.

Capers Jones's books and lectures make a convincing case for increasing the use of statistics in analyzing development activities. However, I suspect that in some of our development organizations we are still mostly asking questions such as, "is it really worthwhile to have a QA organization that focuses on improving the methods of

[19] Wilson, 1990.
[20] Grady and Caswell, 1987; Grady, 1992.
[21] The source of this list is Jones, 1994.
[22] This list comes from a 1993 presentation Jones made to the CQM; it carried a 1990 copyright date.
[23] Ibid.

	Lowest	Median	Highest
1. No design inspections No code inspections No quality assurance No formal testing	30%	40%	50%
5. Formal design inspections only	45%	60%	68%
11. Formal design inspections, and Formal code inspections	70%	85%	90%
15. Formal design inspections, Formal code inspections, and Formal testing	85%	97%	99%
16. Formal design inspections, Formal code inspections, Formal quality assurance, and Formal testing	95%	99%	99%

Figure 11

quality in software development," or "do code inspections fit into our development culture."

Finally, many R&D people have difficulty with the idea of applying statistical methods to R&D because they see R&D as having much more inherent variation than, for instance, manufacturing where they can imagine statistical methods such as SPC applying. In other words, many R&D people consider only the importance of statistical methods such as SPC in minimizing variation, and they do not focus on the necessity of statistical methods such as SPC in detecting whether there is any *significant* variation.[24] If we are going to improve our R&D methods, we must have methods to find the signal in the noise or inherent variation. I am sure that R&D groups are routinely drawing false conclusions from apparent cause-and-effect situations when the apparent effect is really just noise. "Improved" methods based on false conclusions are bound to give improvement activities a bad name in the long run.

I'll finish this subsection with the following quote. In 1883, Lord Kelvin William Thomson said:

> When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginnings of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science.[25]

Focus on Process Goals to Achieve Results Goals[26]

As in every other field of endeavor, if we want to achieve our results, goals, and metrics (e.g., zero software defects), we must concentrate our attention on process goals and metrics (e.g., formal code inspections held). Sports psychologist Jerry May, who works with the U.S. Olympic Sailing Team, explains it clearly:

> I don't think people achieve just by serendipity. If you have visions of what you would like to focus on and accomplish, you have to come up with ways of getting there. I don't want to overstructure goals, but in sailing, if you want to be a great sailor, you need to be thinking about what you want to achieve. We call those "wish goals." The media, the public, and athletes tend to measure sport by those goals. They're okay, but they're overemphasized. Examples are: "I want to win a gold metal" or "I want to be in the top five in this next regatta." Although they may help motivate you, just sitting around wishing about those goals does nothing to achieve them. "Task goals" are the ones that need more attention. These are skill development goals. The sailing coaches and I met, and we came up with six areas where they'd like to see athletes focus on task goals: Sailing technique (such as boatspeed and boathandling), racing technique (starting and tactics), physical conditioning,

[24] Wheeler, 1993.
[25] Bartlett, 1992.
[26] See also Walden, 1994.

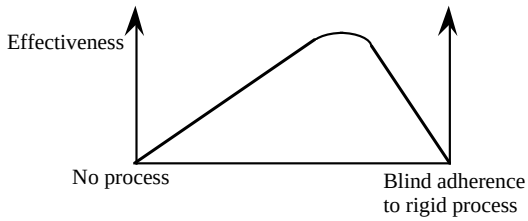

Figure 12

mental skills (such as understanding your arousal level and proper focus), organization (like boat and equipment selection), and personal goals. The last one is a concept I discuss with anyone who is in top-level competition and a high-achiever, whether they are business people, medical students, or athletes. We must find time for personal relationships, education and other kinds of career goals, and recreations.[27]

Make Process More Explicit

We have to make our development processes more explicit. It is impossible to reliably improve a process that isn't explicit. There are several ways of making process more explicit. First, we can build tools or other bits of process technology. In fact, I think one of the greatest sources of potential payoff for R&D departments is investment in process technology. This will be particularly important as we move into the era of mass customization. Second, we can document our existing processes. This is a simple way to make them more explicit and therefore more subject to improvement.

Now, I know that when engineers hear about process they often fear that addition of process will stifle their creativity. I like to illustrate what I think the actual situation is (Fig. 12, above).

On the left side of the horizontal axis we have the point where we have no process, and on the right side we have the point of blind adherence to rigid process—the situation that we in engineering fear so much. The vertical axis is effectiveness. I think that most of us would agree that if we have no process our effectiveness is not going to be very high. We will never do the same thing twice in a row in the same way, no one will ever learn from anybody else, and so on. On the other hand, if we have blind adherence to rigid process, that will also result in low effectiveness; in fact, blind adherence to rigid process is contrary to the idea of continuous improvement. The most efficient operating point is somewhere between the two extremes. This is the point we need to find. I'm betting that in R&D it's to the right of center, as it is in other fields.

We have all seen acts that appear to us to be supremely creative. I believe that in practically every instance, what we take for creativity is actually inspired application of hard-won skill; the renowned dancer/choreographer Martha Graham put it this way: "Technique is the craft which underlies creativity." In many fields—dance, acting, music, debate, language study—improvised performance is practiced until it comes easily, often through reuse of practiced clichés.

Teaching is a good way to discover one's own theory and process

A particularly good way to discover one's own process is to try to teach it. Many of us think we have a theory or process, but in fact we don't have anything that is particularly concrete. However, by having to teach it to other people we have to make it concrete. This lets us sort out what we are actually doing, as well as perhaps suggesting to us things that might be changed.

Embrace Trial and Error

I think also we must embrace the idea of trial and error. Trial and error or experimentation is, after all, the way we find out what works and what doesn't, and it is particularly important to find out what doesn't work and eliminate this from our practices. Embracing trial and error is difficult, however, in many R&D organizations, including some that I have been a part of. We don't feel as if we have the time in business to make any errors. We want assurances that the methods that people try are going to work; thus, we make it dangerous in some cases for people to try new methods and perhaps discover superior ones. Improvement often doesn't result in steadily increasing performance. Rather, performance may decrease at times as one breaks old habits and integrates new skills into the rest of one's body of technique (see Fig. 13, facing page). This is especially true if one is trying to *discover* new methods rather than learn the next skill from the body of proven techniques.

Perhaps one of the reasons "skunk works" organizations work so well is that they allow new things to be tried without letting anybody know they are trying them. I suspect some of the best engineers in your company—I know some of the best engineers in my company—try new things and build new tools without asking anybody; if a new idea doesn't work, they quietly set it aside, but if it does work, they make it available to their

[27] Powlison, 1994.

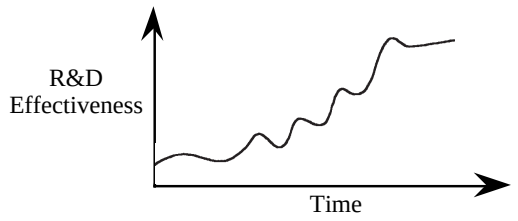

Figure 13

colleagues or throughout the company.

Develop Explicit Theories that Can Be Tested

I think it is not sufficient simply to have processes. We also need theories about what processes should work. To be capable of the greatest improvement, we need to understand why things work. These theories can be derived either by a search of the empirical evidence, or intellectually. In either case, by hypothesizing what might work and documenting it, we put forward theories that can be tested. Notice that the very definition of science is that it deals in theories that can be tested. If somebody puts forth a hypothesis which does not permit testing, that is the realm of religion rather than science. We need to test these theories on individuals, among teams, and among companies. I think at any one time it is healthy to have a number of different theories in practice and test simultaneously. I do not think it is essential or necessarily even healthy that everyone in the CQM, for instance, be practicing Concept Engineering. It is sufficient that some people test the theory that was put forward as Concept Engineering. Other people should be testing other methods. This is the way we go about science.

Now, saying that many theories should be tested simultaneously is not the same as saying that everybody can do their own thing. That does not meet the improvement needs of an organization or of individual employees. Everybody on one team, or in one division, or in one company must participate to test certain overall theories, and if there are differences in individual practices, those must be tested systematically to understand micro-theories.

Publish Theories and Results

I think also that we must publish our theories and results. It is strange to me that, although we have studied science and engineering and benefited substantially from the western tradition of scientific publication and the great technological strides that have resulted from it, many of us are unlikely to publish our own theories and results. It is important to publish so that others may learn from our theories and results or validate them, or so that we can learn from others. Another reason to publish one's theories and results is to force oneself to make the ideas explicit, complete, and consistent, and coherent.

Involve Academics for Improved Validation

I think many companies don't involve academics in a direct way. I think we should actively seek graduate students from our local universities to observe and document our processes, take data, build grounded theory, and then try to validate those theories with other teams. It is not sufficient that we try a lot of new methods; we must also be very sure which methods actually work and which don't. In the press of business we often find it difficult to take the time to do the needed additional analysis that will result in validation or invalidation. The academics can help with this, and we will benefit them by providing realistic situations for them to study and learn from.

Search for Fundamentals

We need more consensus on what the fundamental skills are for excellent R&D performance. In most other fields, including some that are highly complex, like sailboat racing and war fighting, there are skills that are generally accepted as necessary for mastery. Many of us might agree on the fundamentals in some parts of R&D, such as hardware engineering. If so, do we make what these skills are explicit enough to everyone concerned in recruiting, training, and performance? Do we make it clear that we expect them to be mastered, or that we expect a program of development skill development to be in place and functioning on an individual basis? In other areas some people believe they know some fundamentals, such as Jones's four key software development practices mentioned earlier, but many software development groups are still uncertain whether these are really useful methods or whether they can be applied in their own development culture.

I'll put forth a fundamental I believe in (or maybe it's three fundamentals): A large R&D project will inevitably fail unless it has a detailed functional specification, a detailed design, and a detailed implementation plan (with resources actually available to carry out the plan) before implementation starts. In fact, without these three, estimates of project time and cost of product

performance are no more than wishful thinking. Any substantial project inevitably has troubles before it's done. We had better at least have a spec, design, and plan before we start that make success seem very plausible, or the combination of an unplanned implementation and inevitable difficulties will be overwhelming. Now, many of you will say "of course," but I suggest that many times we fall short with regard to this fundamental, and there are many company pressures on us to give short shrift to this fundamental.

Rotate First-rate Practitioners with Passable Pedagogic Skills through a Coaching Role

In R&D organizations, we seldom have coaches. Some managers do it a bit, and some senior engineers do it a bit. However, all too often the people we assign to lead our quality improvement efforts in engineering are not our best engineers, although they may be people with a sincere interest in quality improvement. In many fields the life of a performing professional is relatively short. Thus, a person who has had very great skill is available for much of the rest of his or her life to teach and coach others. In other fields, such as sports or the performing arts, only the world's best performers can make a living performing, and experts who aren't able to earn enough from performing make their living teaching. In engineering, our work life (our performing life) takes up almost all of our career (although it is not clear that all of us are performing at a truly professional level in the later years of our career). Therefore, for the most part our best performers aren't available to teach and coach.

I think we must find some alternative way in R&D organizations to make available coaches who are or were themselves first-rate practitioners. One way would be to rotate first-rate practitioners through a coaching role: for instance, have our best VLSI designer or our best programmer spend the next year (just one year) trying to teach others what they do. This approach would have two benefits, I think: first, the experts would pass on a lot of their knowledge because they would have a lot of time to do it and, second, they would come to understand their own methods better by having to make them explicit enough to teach them to others. I think such a period of teaching would make our best people better.

Another approach might make use of the geography of people's offices or cubicles and work assignments. We talk about providing mentors, but too often the mentor and the new engineer are not seated close to each other. As a result, the mentor relationship becomes one of checking in every once in a while with the new engineer to ask if everything is all right, rather than one in which the mentor takes responsibility for expanding the new engineer's skill. When I first got out of college and went to work at Lincoln Laboratory, I was assigned not to a project but to a senior engineer who was assigned to the project. I was his to use and teach. Soon, offices were rearranged so that I shared the worse half of a two-person office with my mentor. I went everywhere with him and did everything with him. He would break off little pieces of design or implementation (a day or two long) and give them to me to do. We talked through "our" design and implementation together, he answered my questions about why we were doing this or that, and he gently helped me see problems and improvements in what I had done. And he praised me for how fast I did my little bits and how fast I caught on (whether or not it was completely justified).

After a month or two, I was doing bigger pieces, but because we shared an office, my mentor knew when I was struggling and would ask me to talk to him about what I was working on. Also, he would ask me to look over what he had done and give him comments. Since he was responsible for system integration in addition to developing some of the system modules, in time I became the second most knowledgeable person about the whole project. After a year or 18 months, I began to be given more independent assignments, but we still shared the office and he never stopped being my teacher, though I had become his co-worker.[28]

Take Time for Practice Instead of Performance

As I have already mentioned, simply performing a skill often does not result in adequate improvement. We can go for years with the same level of golf game, bridge game, or sailboat racing skill . . . or programming skill, or VLSI design skill. If we are to get better, we must set aside some time for learning new skills and practicing them, or for whipping our old skills back into shape through practice observed by a coach. I firmly believe that the time it would take to do this practice on a periodic basis would be paid for many times over by the per-

[28] After three years I left Lincoln Laboratory and moved to BBN, where I worked as an experienced programmer. Eighteen months later, facing the prospect of having to design and implement the first packet switch for the ARPANET, I was glad when BBN recruited my mentor from Lincoln Laboratory to become my partner (and still my teacher) as we implemented that first packet switch in adjacent offices over the next year.

formance improvements we would then find among our engineers. The trick in R&D, I think—one that will require a good bit of our intellect—is to figure out methods of practicing—to build or refine particular skills.

One way to learn is to volunteer to do other jobs. Since improvement in R&D is so difficult, the more we can learn about what others are doing and the more methods we can learn, the better off each of us will be, I believe. Another method of learning new skills or practicing new ideas is what some people call hacking. When I was a young programmer, we'd stop working on the main job from time to time to code a new tool or simply something that interested us. Sometimes the hack went on in parallel with the main project and sometimes it interrupted the main project. Sometimes the hack resulted in a tool that shortened the overall length of the main project, and sometimes it only provided additional skill or experience. Hacking has a bad reputation in some quarters, and no doubt it has been abused, but it's also a good way to try new techniques and perfect them. When I recruit new developers, I try to learn if they are likely to envision the need for a new tool that will make them more productive and hack it together if it doesn't exist. In the absence of having a good company system for providing time for improvement, I want people who will take initiative to make themselves more productive.

Find Methods to Test One's Form Regularly

Another problem I mentioned in R&D organizations is that our project reviews typically happen only at the end of the project, or perhaps we have a couple of design reviews during the project, but in any case it may be many weeks, months, or even years between reviews that help us understand our level of performance and what we might do to improve it. This is in complete contrast to fields such as sports or the performing arts, where every single day, after every practice, certainly after every match or public performance, there is an exhaustive review of what worked and what didn't work. I think we would benefit greatly in R&D if we tried to provide some method of frequent "after-action review" to help us improve.

In the NIMS case they provided reviews by having the professional sit with a less senior engineer every three or four days and ask him to discuss what problems might arise and to evaluate the state of the design. Then three or four days later the engineer would come back and tell the master engineer whether the problems he anticipated actually happened or not, and whether other problems he had not anticipated had happened; he would also get a review from the master engineer of the quality of his design over the past several days. In engineering, just as in chess or bridge or golf, the non-master usually does not even know whether or not he has a good position or good design. (Of course, in R&D we have a less complete literature of what constitutes a good position or a good design than, say, in the fields of chess or golf.)

People in R&D need to think more about how to find frequent ways to review their R&D skills. NIMS did it by having dozens of small phases rather than five big phases in their phase review process. As I described above, I had the good fortune to have had a master work with me to evaluate my methods and results every few days or hours for four of the first five years of my engineering apprenticeship. Other methods are needed as well.

Proper roles

As discussed under point 13 in section 1, in team endeavors careful manipulation of the participants' roles on the team are very important for team mastery. We need to pay more attention to this area in R&D.

First, we need to find the parallel in R&D to the idea practiced by many successful teams in other fields, in which the individual team members must be primarily committed to the success of the team, and their individual success comes from being members of a successful team from which they can gain experience and new skills.

We also need to help team members understand the benefits to themselves and the team of "playing different positions" over time. For instance, in software development, test activities and development skills would be improved if developers rotated through testing periodically and vice versa. However, developers often look down on the test function and resist spending time there, and test people themselves are often uncomfortable about the idea of having to try a rotation through development.

Some teams members are *a lot* better than others and can personally make an enormous difference in team performance. In instances like this, if great team performance is needed quickly (for instance, because the product release date is coming fast), it may be important for some participants to concentrate on helping the best players maximize their output rather than all participants working as equals. In other cases, it may be necessary for one of the better people to

temporarily take on a "lower" or support assignment for the good of team performance.

A strong team leader needs to know how to arrange for people to play the roles that best support the team as circumstances vary over time. Individual members who willingly seek to perform, with all their energy, whatever role is most important to the team at any time are always popular with team leaders and team members throughout an organization, and such people can have a great impact on team success.

What About Natural Talent?

Up to now I've avoided the issue of natural talent for R&D and made the point that most of us, regardless of our level of natural talent, have plenty of room for improvement with appropriate training, practice, and discipline.

Natural talent for an activity is never enough alone to achieve mastery. Mastery still requires lots of training and practice. However, natural talent for an activity may help one go farther faster. There may be a natural "mental physique" for various R&D activities, just as there is a natural "physical physique" for various athletic activities. To perform at the highest levels one must develop this physique (no one is ever born with it completely developed). Thus, the R&D world (although maybe not just one company) might use its scientific discovery methods to identify the essential skills for the mental physique of a champion R&D person. Identifying these skills would allow us to accurately spot people with these abilities, help develop such abilities in those who have them innately but as yet undeveloped, and find methods to help people compensate who lack such abilities. Such compensatory methods might involve developing the key skills through hard hard work, providing alternative skills to serve in place of the key skills, or organizing teams so someone else on the team has the needed ability.

What About Finding Time?

The most frequent objection to improvement efforts in R&D is "we don't have time." However, we must take the time to do such things as I have mentioned, or find ways to do them that don't take extra time (such as the master and apprentice sharing an office). We must take time to visit customers. If we don't, we will waste more time developing products that don't directly meet customers' real needs, and we will waste time trying to discover innovations that customers could have just given us. (A way to do this that doesn't take time might be to invite people from the customer organization to join our development team.) We must take time to do frequent reviews of our results and methods. If we don't, we will waste more time using suboptimal methods. We must take time to discover valid metrics for our results and methods. If we don't, we will waste time using poor or possibly counterproductive methods. Every bit of time we waste and every mistake we make because we haven't taken the time to improve our skills allows competitors who are more skilled than we are or working harder at improvement than we are to close the gap and to pass us.

If we can't find a way to undertake improvement activities without taking additional time, we simply have to find ways to take the time. Sometimes the payback will be relatively immediate, and we either "know" this ourselves or there is much extant evidence of certain payback. For instance, taking a week or two now to make a big program follow some rigid portability guidelines or intermodule communication conventions will surely pay for itself by the time we *actually* get the current release out, even if it *apparently* delays the scheduled release date; and it will continue to pay for itself with big multiplicative factors over the life of a product.

One of the problems we face is making time for team meetings to review improvement results and plan new improvement activities. There is always something more pressing to do, for example, responding to a customer complaint, or providing something to someone else to keep from holding him up. Many times I have heard engineers say, "I'm already working infinitely hard; there's no time to add improvement work." My thought is that the proper place for team improvement work is Monday from 8:30 to 10:30 a.m. Then it's out of the way for the week (and we can be trying the new ideas), and we can still fit in nearly as much "real" work in the rest of the week.

Much improvement work is individual and doesn't require team meetings, however. We didn't tell anyone we were taking time to reformat all the comments and indenting in our program to use the conventions we now believe are best. Much improvement work can be done incrementally, without disturbing anyone, if we ourselves believe it is important. Mastery takes a lot of time over a long time. As the Zen master told the young student, "if it takes a long time to do, we must begin today."

Motivation

Improving ourselves takes time, and it takes dis-

cipline. In other words, improvement is hard. It will never happen if we aren't motivated to do it. A friend who is a champion sailor lists all the systematic improvement methods that were essential to making him a champion. However, first on his list was the urgency he had about improving and the importance he placed on winning, which gave him the motivation to do the practice and skill building necessary to win. Some of this motivation can be provided by our companies (certainly they can work to remove obstacles to motivation). Some of it can be provided by our managers or leaders; and, in fact, we often define a good leader as one who can provide such motivation. However, ultimately, over the long run, each of us mostly does what we want to do. If we aren't motivated to do improvement work, we will manage to avoid it. As Edgar Schein says, "you can't motivate change; you can only try to link existing motivation."[29, 30] We can stymie improvement efforts in R&D if we want to, and then sit back and complain about the company, its management, and their lack of success.

Ultimately the issue is what we individually define our jobs to be. If we define them to include self-improvement, then we will be noticed by and will gravitate to others who are also trying to improve themselves, as well as to team leaders who are selecting people for the teams that are going to be competitively dominant; we can then be part of the solution rather than part of the problem. The source of our motivation is not important; it may be desire to win, desire to become proficient, desire to be a team member, desire for recognition, or desire for self-sacrifice. In every field of team endeavor, there are expert performers (winners) who are on the lookout for people willing to commit to the team goals and work enthusiastically to improve themselves to serve the team better.

A Culture of Learning and Improvement

When we try to improve skills in an R&D organization, we often speak of having to change the culture (because we start in most cases from R&D cultures where skill improvement is not a primary ingredient). I sometimes wonder if it might be easier to start from scratch, e.g., at a new start-up where one selects new hires partially because of their desire to participate in a culture of learning and improvement, or to start a new R&D division in a company where one condition for joining the new division is being interested in living in a culture of learning and improvement. I am convinced someone somewhere is creating a division or company based on much greater mastery in R&D and this will become a business strategy that can be used to dominate one's competition. If we have an existing division and decide to have an environment of learning and improvement, I sometimes wonder how much cultural charge one can succeed in producing in people who don't want to change. Perhaps we have to make the staff decide whether to participate or leave. The successful basketball coach or sailboat captain does not have much time for people who aren't willing to commit to the program. Still, I suppose there are instances in R&D and elsewhere where groups manage to change to a culture of learning and improvement.

There is an interesting parallel between the difficulty businesses have in getting their employees to improve themselves and the difficulty our public schools have in getting our kids to learn. In general, both types of institutions are getting people to learn new things. In a surprising number of cases where schools are successful in getting kids to learn, the schools or the parents have managed to provide an environment of study and an expectation of learning. For instance, the parents of refugee school children from Southeast Asia who are disproportionally successful in school "set standards and goals for the evening [homework] and facilitate their children's studies by assuming responsibilities for chores and other practical considerations. Older siblings help their younger siblings. Indeed they seem to learn as much by teaching as from being taught."[31] Parents expect their children to surpass them academically. In turn, schools which succeed tend to provide a rich environment for study, personal monitoring by interested teachers, and an expectation that the children will learn. In other words one of the solutions to the difficult problem of motivating the will to learn and to improve is to be in a culture of learning and improvement.

One can speculate on how much improvement would happen in an R&D organization if it provided a culture of learning and improvements, e.g., if engineering managers set aside regular time for study, if managers relieved workers of chores so the workers could study, if more experienced workers (or managers) taught less experienced workers, if the business "family" was determined to train its young workers,

[29] Schein, 1987 and 1988.

[30] Some systems of training explicitly seek to discover who has this intrinsic motivation and to develop it and to excuse from participation whose who don't have it. See for instance, Donnithorne, 1993. One of the ways people traditionally motivated themselves is to participate in study or practice groups which involve difficult to break commitments to one's study and practice partners.

and if managers had the goal of having the next generation surpass them. I think it would make a big difference.

Conclusion

I'd like to conclude with the following summary.

I believe that achieving mastery of one's field is the key to competitive success. In addition, mastery is gratifying in its own right—the continuing learning that is required to gain mastery is one of life's great joys. The best people don't quit jobs because there is too much happening; they quit jobs and move on when they are not learning anything new. Increasingly, mastery of one's field and lifelong learning are required both for the success of our companies and for our individual success in our careers.

We will be much better off if we take control of our own destiny and do the necessary improvement work in our R&D organizations (or if necessary by ourselves individually) than if we yield that responsibility to someone else, who perhaps doesn't understand scientific improvement methods. It is a mistake, in my view, for an engineer to say that it is the company's responsibility to help him improve his skills. It is certainly in the company's interest to help engineers improve their skills, but ultimately it is the engineers' responsibility to improve their own skills. [This is particularly true as we move toward the world of the virtual corporation where an increasing number of engineers may be essentially self-employed.]

TQM is a set of methods for the systematic development of skill at the individual, team, company, and societal level for succeeding in business in a rapidly changing world. Thus, TQM is a mechanism that empowers those of us in R&D to improve our own skills. We are being asked to do what is necessary to improve the skills of our organization, of our teams, and of ourselves. This is not the time to be reticent; it is the time to embrace that empowerment and get to work.

There is no one more qualified than we in engineering to figure out how to apply quality methods to our situation. But it is a very complex situation, and it is going to take all of our intellect and enormous energy. We need to attack the issue of how to become more productive and more competitive, as individual workers, as teams, and as R&D organizations systematically (using scientific methods). It will help our companies, and we owe it to ourselves—somewhere there is a group of people who are already working harder and more efficiently than we are to develop their skills, and we could be their target. To paraphrase Donald Wheeler, "people who understand scientific methods of improvement and don't use them will have no advantage over those who can't use them."

Thank you very much. ■

Acknowledgments

As I was preparing to draft this paper, I talked either at length or briefly to golfers Ean Rankin and Jonathan Crane, tennis players Harry Kirsch and Mike Nacey, chess master Alex Cherniack, champion sailor Stewart Neff, sailing coach Ken Legler, violin teacher Ed Pearlman, management consultant Jack Reilly, and dancer Michael Grandfield. Gary Burchill, Tad Elmer, David Ford, Dan Friedman, Alan Graham, Michael Grandfield, Tom Lee, Noel Mulcahy, Jack Reilly, Rick Schantz, Al Sciacca, Ted Walls, and Ed Walker also read and commented on drafts of the paper. Astrid Delori and Deborah Melone helped with preparation of the manuscript. I greatly appreciate the insight and help I received. I also always remember with gratitude the lessons from the years I worked closely with my software mentor, Will Crowther.

[31] Caplan et al., 1992.

Bibliography Relating to Development of Skill and Improving the Product Development Process

Donald J. Albers and G. L. Alexanderson, editors, 1985. *Mathematical People: Profiles and Interviews,* Birkhauser, Boston, Massachusetts.

John Bartlett, 1992. *Familiar Quotations: A Collection of Passages, Phrases, and Proverbs Traced to their Sources in Ancient and Modern Literature,* 16th edition, Little, Brown and Company, Boston, Massachusetts.

W. I. B. Beveridge. *The Art of Scientific Investigation,* Vintage Books, New York.

Theobald Boehm, 1964. *The Flute and Flute Playing in Acoustical, Technical and Artistic Aspects,* Dover Publications Inc., New York.

H. Kent Bowen, Kim B. Clark, Charles A. Holloway, Dorothy Leonard-Barton and Steven C. Wheelwright, 1994. "Development Projects: The Engine of Renewal," "How to Integrate Work and Deepen Expertise," and "Make Projects the School for Leaders," three papers in a special section on "Regaining the Lead in Manufacturing," *Harvard Business Review,* Vol. 72, No. 5, September-October Issue, pp. 108-140.

Vic Braden and Bill Burns, 1977. *Vic Braden's Tennis for the Future,* Little, Brown and Company, Boston, Massachusetts.

Gary W. Burchill, 1993. *Concept Engineering: An Investigation of TIME vs. MARKET Orientation in Product Concept Development,* MIT, Doctor of Philosophy Thesis.

Nathan Caplan, Marcella H. Choy and John K. Whitmore, 1992. "Indochinese Refugee Families and Academic Achievement," *Scientific American,* February Issue, pp. 36-42.

Gay Cheney, 1989. *Basic Concepts in Modern Dance: A Creative Approach,* Princeton Book Company, Princeton, New Jersey.

Dennis Conner with John Rousmaniere, 1987. *No Excuse to Lose,* W. W. Norton & Company, New York.

Stan Davis and Jim Botkin, 1994. "The Coming of Knowledge-Based Business," *Harvard Business Review,* Vol. 72, No. 5, September-October Issue, pp. 165-170.

Stanley M. Davis, 1987. *Future Perfect,* Addison-Wesley, Reading, Massachusetts.

Larry R. Donnithorne, 1993. *The West Point Way of Leadership,* Currancy Doubleday, New York.

Peter F. Drucker, 1985. *Innovation and Entrepreneurship: Practice and Principles,* Harper & Row, New York.

Anders K. Ericsson and Neil Charness, 1994. "Expert Performance: Its Structure and Acquisition," *American Psychologist,* Vol. 49, No. 8, August Issue, pp. 725-747.

David Falkner, 1990. *Nine Sides of the Diamond: Baseball's Great Glove Men on the Fine Art of Defense,* Times Books, New York.

Richard Florida and Martin Kenney, 1990. *The Breakthrough Illusion*: *Corporate America's Failure to Move from Innovation to Mass Production,* Basic Books.

John Gardner, 1985. *The Art of Ficton: Notes on Craft for Young Writers,* Vintage Books, New York.

Michael J. Gelb and Tony Buzan, 1994. *Lessons from the Art of Juggling: How to Achieve Your Full Potential in Business, Learning and Life,* Harmony Books, New York.

W. Wayt Gibbs, 1994. "Software's Chronic Crisis," *Scientific American,* Volume 271, Number 3, September Issue, pp. 86-95.

Steven K. Goldman, Roger N. Nagel, and Kenneth Preiss, 1994. *Agile Competitors and Virtual Corporation: Strategies for Enriching the Customer,* Van Norstrand Reinhold.

Robert B. Grady, 1992. *Practical Software Metrics for Project Management and Process Improvement,* Prentice Hill, Englewood Cliffs, New Jersey.

Robert B. Grady and Deborah L. Caswell, 1987. *Software Metrics: Establishing a Company-Wide Program,* Prentice Hill, Englewood Cliffs, New Jersey.

Gary Hamel and C. K. Prahalad, 1991. "Corporate Imagination and Expeditionary Marketing," *Harvard Business Review,* July-August Issue, pp. 81-92.

Charles Handy, 1990. *The Age of Unreason,* Harvard Business School Press, Boston, Massachusetts.

Cornelius Herstatt and Eric von Hippel, 1992. "From Experience: Developing New Product Concepts Via the Lead User Method: A Case Study in a 'Low-Tech' Field," *Journal of Product Innovation Management,* Vol. 9, pp. 213-221.

Watts S. Humphrey, 1987. *Managing for Innovation: Leading Technical People,* Prentice-Hall, Englewood Cliffs, New Jersey.

Capers Jones, 1991. *Applied Software Measurement: Assuring Productivity and Quality,* McGraw Hill, New York.

Capers Jones, 1994. *Assessment and Control of Software Risks,* Yourdon Press, Englewood Cliffs, New Jersey.

Kevin Jones and Tatsuo Ohbora, 1990. "Managing the 'Heretical' Company," *The McKinsey Quarterly,* No. 3, pp. 20-45.

Roger Kahn, 1972. *The Boys of Summer,* Harper & Row, New York, (see, for instance, the chapter on George Shuba).

Robert Keidel, 1985. *Game Plans: Short Strategies for Business*, E. P. Dutton, New York.

Arthur L. Keigler II, 1992. *Improving Product Development by Improving Coordination*, MIT, Master of Science Thesis.

Robert Kelley and Janet Caplan, 1993. "How Bell Labs Creates Star Performers," *Harvard Business Review*, July-August Issue, pp. 128-139.

Donald E. Knuth, 1992. *«Literate Programming»*, Center for the Study of Language and Information, Leland Stanford Junior University, Chapter 10, "The Errors of T_EX (1989)," pp. 243-291, Chapter 11, "The Error Log of T_EX (1978-1991)," pp. 293-339.

Fumio Kodama, 1991. *Analyzing Japanese High Technologies: The Techno-Paradigm Shift*, Pinter Publishers, New York.

Alexander Kotov, 1971. *Think Like a Grandmaster*, Chess Digest, Dallas, Texas.

John Krell, 1973. *Kincaidiana: A Flute Player's Notebook*, Trio Associates, Malibu, California.

Steve Maguire, 1993. *Writing Solid Code*, Microsoft Press, Redmond, WA.

Steve McConnell, 1993. *Code Complete: Practical Handbook for Software Construction*, Microsoft Press, Redmond, Washington. (See Chapter 33 for the author's recommendations for the library for a software professional.)

Michael E. McGrath, Michael T. Anthony and Amram R. Shapiro, 1992. *Product Developement: Success Through Product and Cycle-time Excellence*, Butterworth-Heinemann, Boston.

John McPhee, 1978. *A Sense of Where You Are: A Profile of Bill Bradley at Princeton*, Farrar, Straus and Giroux, New York, second edition.

Marshall Miles, 1959. *How to Win at Duplicate Bridge*, Exposition Press, New York, third edition.

Andre Millard, 1990. *Edison and the Business of Innovation*, The Johns Hopkins University Press, Baltimore, MD.

Jack Nicklaus with Ken Bowden, 1974. *Golf My Way*, Simon and Schuster, NewYork.

B. Joseph Pine II, 1993. *Mass Customization: The New Frontier in Business Competition*, Harvard Business School Press, Boston.

Dave Powlison, 1994. "Psyching for Sailing," *Sailing World*, Sailing Medalist Section, July Issue, pp. 14-18.

C. K. Prahalad and Gary Hamel, 1990. "The Core Competence of the Corporation," *Harvard Business Review*, May-June Issue, pp. 79-91.

Bill Russell and Taylor Branch, 1979. *Second Wind: The Memoirs of an Opinionated Man*, Ballantine Books, New York.

Edgar H. Schein, 1988. *Process Consultation: Its Role in Organization Development*, Volume I, Addison-Wesley Publishing, Reading, Massachusetts.

Edgar H. Schein, 1987. *Process Consultation: Lessons for Managers and Consultants*, Volume II, Addison-Wesley Publishing, Reading, Massachusetts.

Shoji Shiba *et al*., 1993. *A New American TQM*, Center for Quality Management, Cambridge, MA and Productivity Press, Portland, OR.

Preston G. Smith and Donald G. Reinertsen, 1991. *Developing Products in Half the Time*, Van Nostrand Reinhold, New York.

David Sudnow, 1978. *Ways to the Hand: The Organization of Improvised Conduct*, Harvard University Press, Cambridge, Massachusetts.

Sheridan M. Tatsuno, 1990. *Created in Japan: From Imitators to World-Class Innovators*, Harper.

William Taylor, 1990. "The Business of Innovation," *Harvard Business Review*, March-April Issue, pp. 97-106.

Eric Twiname, 1982. *Sail, Race and Win*, Sail Books, Boston, Massachusetts; also Eric Twiname and Revised by Cathy Foster, 1993. *Sail, Race and Win*, Second Edition, Sheridan House, Dobby Ferry, NY.

Kiyoshi Uchimaru, Susumu Okamoto, and Bunteru Kurahara, 1993. *TQM for Technical Groups: Total Quality Principles for Product Development*, Productivity Press, Portland, Oregon.

Various authors, 1994. "Metrics of Software," *Computer* special issue, IEEE Computer Society, Vol. 27, No. 9, September Issue, pp. 13-79.

Eric von Hippel, 1986. "Lead Users: A Source of Novel Product Concepts," *Management Science*, Vol. 32, No. 7, July Issue, pp. 791-805.

Eric von Hippel. *Managing Innovation Through Lead Users*, Video Course, MIT Center for Advanced Engineering Study.

Eric von Hippel, 1988. *The Sources of Innovation*, Oxford University Press, New York.

David Walden, 1993. "Breakthrough and Continuous Improvement in Research and DevelopmentæAn Essay," *The Center for Quality Management Journal*, Vol. 2, No. 2, Spring Issue, pp. 25-29.

David Walden, 1994. "Thoughts on Goals and Metrics," *The Center for Quality Management Journal*, Vol. 3, No. 1, Winter Issue, pp. 33-38.

Stewart H. Walker, 1980. *Winning: The Psychology of Competition*, W. W. Norton & Company, New York.

Donald J. Wheeler, 1993. *Understanding Variation: The Key To Management Chaos*, SPC Press, Inc., Knoxville, Tennessee.

Edith Wilson, 1990. "Product Definition:

Assorted Techniques and Their Marketplace Impact," *1990 IEEE International Engineering Management Conference*, pp. 64-69.

Edward O. Wilson, 1994. *Naturalist*, Island Press, Washington, DC.

James P. Womack, Daniel T. Jones, and Daniel Roos, 1990. *The Machine that Changed the World: The Story of Lean Production*, Harper Perennial.

Edward Yourdon, 1992. *Decline and Fall of the American Programmer*, Yourdon Press, Englewood Cliffs, New Jersey.

David B. Youst and Laurence Lipset, 1994. "Survival of the Fittest," *IEEE Spectrum*, Volume 31, Number 10, October Issue, pp. 67-69.

The Center for Quality of Management Journal is a forum for disseminating the experience of organizations learning to implement modern management practices. It seeks to capture experiences and ideas that may be useful to others working to create customer-driven, continuously improving organizations.

The CQM Journal is refereed. However, it is not an academic publication. Experiences and ideas will be published if they seem likely to be useful to others seeking to improve their organizations.

Send to:

The Center for Quality of Management Journal
Editorial Department
One Alewife Center, Suite 450
Cambridge, MA 02140
Tel. 617-873-8950 Fax 617-873-8980
E-mail: publications@cqm.org

If you have thoughts for a paper and you would like to discuss it with us, please write, call or submit an outline. We welcome your ideas.

Final Manuscript Requirements:

Entire manuscript should be double-spaced, including footnotes, references, etc. Text should include all the elements listed below. Generally, The CQM Journal follows the editorial principles of The Chicago Manual of Style. We strongly prefer submissions in eletronic format for all text and as many of the figures as possible. IBM-based software (particularly Microsoft Word for Windows) is preferable to Macintosh-based software if you have a choice, but is by no means a requirement.

Please include:

1. Title page, stating the type of article (e.g., 7-Step case study, research paper, short communication, letter to the editor, etc.), main title, subtitle, and authors' full name(s), affiliation(s), and the address/phone/fax of the submitting author;

2. All needed figures, tables, and photographs (see below);

3. Footnotes (if appropriate), numbered consecutively from the beginning to the end of the article;

4. Reference list, if appropriate.

Figures, Tables and Photographs:

If you can, insert each figure or table into the text where you would like it to fall. Figures should be composed to conform to one of two widths: 3 1/8 or 6 1/2 inches. The maximum height for any figure is 9 3/8 inches. Text within figures should not be smaller than 5 points and lines not less than 1/4 point at the figure's final size. Figures should labeled with the figure number underneath and title on top. Be sure that the text mentions each figure or table.

Please retain separate PICT or TIFF files of figures generated in drawing programs and a file with the text only for final submission.