

The PIT Crew

A Grass-Roots Process Improvement Effort

Danny R. Faught

faught@asqnet.org

This is the saga of the PIT Crew. PIT is an acronym for Process Improvement Team, and yes, “PIT Crew” is redundant, but we like the sound of it. PIT is a grass-roots process improvement effort with a genesis that can be traced back to 1994. The PIT Crew is comprised of members of what is now the Integration, Test, and Delivery team (ITD) in Hewlett-Packard’s Richardson System Software Lab in Richardson, Texas. PIT has survived several reorganizations, management changes, and a merger. It has become a part of the ITD culture, but it continues to evolve. It will be a useful exercise for us to look back on PIT’s history, and perhaps others will benefit from riding along on the roller coaster with us.

I’ll start by describing two different efforts that lead to PIT’s creation. I’ll detail the training and early tasks that PIT undertook, and then I’ll summarize some of the projects that PIT has completed. I’ll describe how PIT has evolved and compare PIT to other improvement teams. I’ll close by revealing how we have come full-circle back to our genesis and I’ll speculate on how we may evolve in the future. References appear at the end.

Some background will make the journey more understandable. In 1994, we were the System Software Test Group within Convex Computer Corporation, the company that created the “minisupercomputer” niche in the computer industry. We were responsible for all facets of test development for our operating system software. We were part of a larger operating system development organization. Late in 1995, Convex was acquired by Hewlett-Packard, and some time after that ITD broadened its responsibilities and became the Integration, Test, and Delivery group, taking responsibility for the first round of software integration and shedding some functional testing duties. When I mention “the test group,” I am referring to the various incarnations of the System Software Test Group and ITD. The PIT Crew is a subset of this group.

Roots

We can trace the roots of PIT back to a project called “Stuntman” which we started in early 1994. Stuntman got its name from the fact that “it would drive anything”. The test group wanted to replace the user interface for our test harness, and also wanted to expand the user interface to handle other test tools so that they would all have a consistent interface. Our existing home-grown user interface had become unmaintainable due to its continual feature creep, so adding new functionality was difficult. We eventually picked up representation from another local test group that used the same test harness, and our local tools group.

I'm pretty embarrassed as I read over my status reports from this period. I was trying to lead a software development project, having only been in the workforce for just over a year after graduating with a Bachelor's degree in Computer Science. I had very little knowledge of software engineering life-cycles, and I had no idea how to estimate the calendar time or effort for a project. I'll have to save my gripes about Computer Science degrees for another paper. Suffice it to say that I had to readjust my schedule many times, changing both the delivery date and what I was promising to deliver.

Later, a study group was forming to learn object-oriented analysis and design, and we decided to use Stuntman as the class project. The study group proceeded to meet weekly with a few of the people on the Stuntman project in attendance, and the full Stuntman team also met regularly in order to apply the OO techniques we were learning and do some more involved analysis than was appropriate for the study group. With this motivation, we made steady progress for a few months, but we soon realized that devoting an hour a week to the project would not be sufficient. We started meeting for two hours or more, once or twice a week. We completed the requirements specification to our satisfaction, and started work on the design. Another month after that, I realized that my main job responsibilities were not going to change to accommodate the work on Stuntman as a skunkworks project. I passed the leadership on to a Stuntman team member, who was a member of the development organization's internal tools group.

The Thrashings Will Now Begin

To be more accurate, the thrashing got worse. Any time a new leader comes in, an adjustment period is inevitable. We started rethinking the design we had so far. Most meetings degenerated into answering the question, "Why did we decide to do this project in the first place?" Our new leader weathered the storm admirably, but somehow we didn't give him much to work with. Despite the progress we were making on the design, support from the team was crumbling. "Stuntman" became a dirty word, and to some old-timers, it still is. We referred to "the S-word" project.

The Stuntman project fell by the wayside around the time of the merger with Hewlett-Packard at the end of 1995. There are many factors that contributed to this. We had a deadline that we wanted to meet, but there were no serious consequences for missing that deadline. We hadn't learned to push back on other schedule pressures, which might have warned us earlier that we had a lack of management support with respect to people resources. Also, we had never tried a reengineering effort of that magnitude relative to the number of players. To this day, we shy away from large improvement projects unless we can divide them into incremental chunks with tangible and useful deliverables at each step. And of course, the merger, which was nonetheless executed very well, occupied our attention.

Still a Spark...

Through the winter of 1996/1997, our desire to improve our user interface was still strong. We decided not to replace our existing software entirely, but rather to find the least painful way of improving what we had or to replace parts of it. In the spirit of Stuntman, we wanted to call this effort "Fall Guy," but an unfortunate pizza overdose resulted in the team choosing the name "Lee Majors Fan Club" (LMFC). A former Stuntman team member filled the leadership role. This team initiated a number of improvements to our existing user interface software, some of which were successful, and others of which we eventually abandoned. We built up some momentum with our success, and by early summer 1997, I sought to expand LMFC's charter into a full-blown process improvement team, and within a month, the transition was complete. PIT was born. For unrelated reasons, the LMFC team leader moved on, so the new Process Improvement Team got a new leader, recruited from the test group.

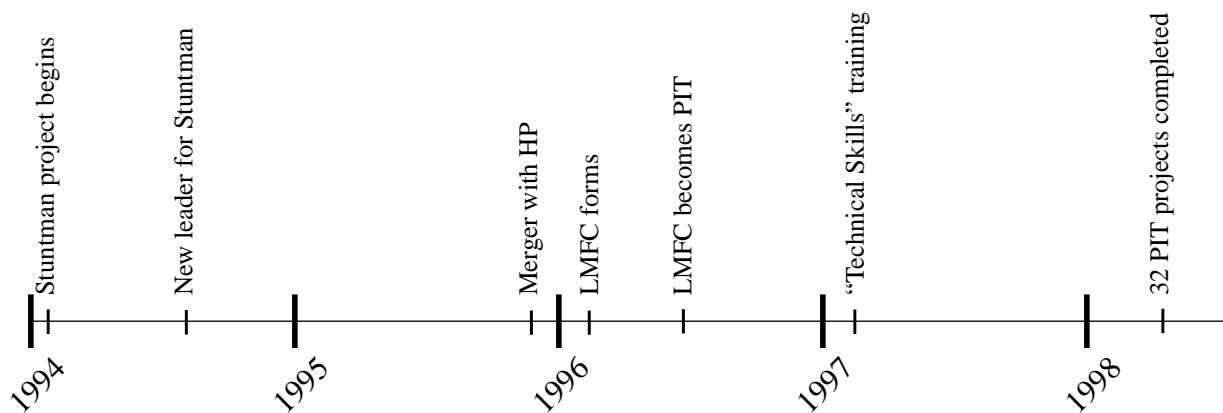


Figure 1. An approximate time line for the PIT crew, based on some electronic archaeology.

Training

As soon as the PIT Crew officially formed, I looked for opportunities to provide training. I distributed copies of a couple of articles from *Software QA Magazine*—"The Quality Improvement Core Team: A Grass-Roots Approach to Process Improvement" [DENT96], and "Jump-Start Your Stalled Improvement Effort" [ALEXANDER96]. I had joined the American Society for Quality, and I occasionally shared information from the ASQ's *Quality Progress* magazine.

I had been sponsoring a series of "Technical Skills Sessions" where we shared our knowledge of relevant topics with each other. A few months after PIT's creation we decided to dedicate one of these sessions to the Quality movement. We recruited three people to read different books—*Out of the Crisis* [DEMING86], *Quality is Free* [CROSBY79], and *Quality is Still Free* [CROSBY96]. At the technical skills session, each volunteer presented a book review and shared the major points that they felt were relevant to us. We also recruited the "Quality Guy" at our site to give an overview of the quality movement (unfortunately, his priorities didn't allow for much further interaction with the software development groups, though his group did expand later to provide a little more attention to software issues). We set up a teleconference and invited a few testers and software QA people from other HP sites.

The feedback from this session was positive, and we even had a few managers attend. However, we didn't give people enough information to be able to apply much to their daily work, besides a few general principles. I later attended the "First Pass" training class sponsored by HP where I learned some of the basic quality tools, such as the fishbone diagram and Pareto charts. This was a valuable experience. I experimented with the techniques for root cause analysis after a project postmortem and got tangible and positive results. I'm still working on folding the use of some of the techniques I learned into the basic makeup of PIT's procedures.

A few members of the PIT Crew have since been pushing for more quality training for the group, but we had to shelve the idea due to lack of interest. This is somewhat disturbing, knowing that in Japanese factories, it is typical for all line workers to be trained in statistical quality techniques. However, the process improvement training that we seek is more relevant to meeting facilitators and team leaders than individual engineers. We haven't yet defined metrics for individual contributors to track their own work. We're also struggling somewhat trying to translate the various quality techniques into something relevant to a software engineering process. At a recent lecture I attended

[STIMSON98], the speaker stated that a cost/benefit analysis is important, and I asked how this could be applied to a software process where the benefits are often very hard to measure. He gave the typical consultant's response, that he'd have to find out much more detail about what my group was doing before he could help. We need to find quality consultants and other resources that already understand and account for the general workings of a software engineering process.

With a little bit of networking, I was able to find several local organizations that group members could benefit from, both in the area of process improvement as well as other areas relevant to the job [see ORGS in the References section]. These organizations typically offer monthly meetings as well as special interest groups or tutorials. They sometimes also offer specialized workshops that are cheaper and more accessible than a national conference or remote training. See the resources section for details. We're lucky to be located within a large metropolitan area. Companies that aren't so centrally located have to be more creative in order to take advantage of the available organizations without requiring hours of commuting. It's hard enough to convince a group member to spend an evening or weekend to enhance their career. Adding travel time makes it more difficult. Luckily, there is a wide array of online resources that doesn't discriminate based on geography [see the ONLINE section in References].

Early Tasks

We had a number of tasks at hand as we cranked up the activities of the PIT Crew. First, though we could identify a few individuals with particular interest in PIT, we never defined an official membership roster. We allowed people to come to the meetings when they had the time or a particular interest in a PIT project. We decided that all communication would go to the email alias for the entire test group, with the subject starting with "PIT" in case someone wanted to filter it out. We wanted the entire test group to feel like they were a part of PIT, and we wanted them to understand that PIT's activities affected us all.

Tracking

Our first substantive task was to decide how to track improvement suggestions and ongoing projects. We defined our requirements, considered a few alternatives for a tracking system, and decided to use the same tool that we were using to track bug reports (Viper, a tool developed internally by our Information Technology group). Viper was not a perfect fit, especially in terms of project management, but it was readily available and we already knew how to use it. We set up a module in Viper to receive process improvement suggestions separately from software bugs. We're quite fond of the paradigm of submitting "bugs" against our processes. We do some of our project tracking outside of Viper, mostly in the form of an action item list that we carry forward from one meeting to the next.

We use the bug priority field to indicate whether a project is active (A), new (B), waiting in the wings (C), and interesting but not terribly relevant (D). The default priority for incoming bugs is "B." We chose to use that priority to indicate new suggestions that we hadn't analyzed yet, so we could tell everyone submitting suggestions to simply use the default priority. It's appropriate to give this fairly high priority to the task of analyzing new bugs, at least until we have thought about them enough to update the priority. We do not allow a bug to stay at priority B once we've analyzed it. We're willing to accept the slightly strange progression, A-C-D, for bugs already in process, because we have more control over that part of the process.

When we realized that some bugs that came in were not well-written and we were having to consult the authors for clarification, we decided to create a bug template:

<Delete all help text in angle brackets. See our web page for an example.>

Problem: <State the problem briefly in one or two lines, starting here. Be as specific as you can in two lines.>

Impact: <State the negative effects that the problem is causing.>

Detailed Description: <Explain the problem in detail.>

Proposed Solution: <Optional. If you have any solutions in mind, be sure to document them in detail.>

Pre-analysis: <Optional>

- S - Is the problem Specific?
- M - Is the solution Measurable?
- A - Is the solution Achievable?
- R - Is the problem Relevant?
- T - Is the solution Trackable?

Additional Information: <Any other relevant information>

Previous: <If this is a followup to a bug already in the database, provide the bug number here>

Next: <Before you close a PIT bug, open a new one for the next logical step and add the bug number here>

Children: <If this is a high-level bug related to a more specific bug, specify all the lower level bugs here>

Parent: <If this bug is related to a higher-level bug, provide the parent's bug number here>

This template has been useful for keeping suggestions in a standard form. Note that the items in the template are all contained within the description portion of the bug report. Since we're able to do keyword searches on the description field, we haven't had a need to promote the various parts to separate fields in the relational database, and since most Viper users don't use it for process improvement tracking, it would be difficult to justify any customization.

The Problem field is intended to provide a summary. This field is sometimes the only part of the description that we print out when we're reviewing a stack of bugs. The Impact field is very useful for reminding people to clearly explain exactly why something is a problem. We ask for as many details as possible in the Detailed Description field, and we give submitters the option to suggest a solution or two in the Proposed Solution field. It's very important to separate the description of the problem from the proposed solution. We don't want to get into "genius mode" where we think it's obvious that a particular solution needs to be implemented. It's common that PIT will choose a completely different solution once we've analyzed the problem. Even when the solution is presented separately, and especially when it isn't, there is a danger that the bug will be rejected because the PIT participants don't like the solution. This is unfortunate when it happens, because it leaves the underlying problem unresolved. See [TAMMANA98] for a more thorough discussion on the pitfalls of bug reporting.

You'll see in the Pre-analysis field that we borrowed the "SMART" paradigm from our earlier training on goal-setting. This sort of analysis is a good idea in general, but we found that it was not effective to ask the person originally writing up the suggestion to do this analysis. It's better for the process improvement team to do this along with their usual analysis of the suggestion. We occasionally used the Previous, Next, Children, and Parent fields to split a broadly-scoped bug into separate bite-sized projects and to track followup projects after one was completed. It was useful for someone closing out a project to seriously consider what additional improvements could be made and to open this as a new bug as part of the process of closing out the previous one. This also allowed us to get closure on a project by putting as yet uncompleted tasks into a new bug.

We sometimes wondered whether an item that landed in our bug database really concerned a process was simply a bug in some existing test software that the test group supported. Using the same track-

ing system for software bugs and process bugs contributed to this. But it also made it simple to move bugs from one module to another. A major enhancement request is likely to be moved to PIT's Viper module, while a bug in some existing software feature will be moved to the module for that software.

An important part of our PIT bug tracking is a periodic review of the bugs at the various priority levels. Priority A bugs are active projects, so they are tracked at every meeting. Incoming priority B bugs are analyzed at most PIT meetings. We defined a rigid interval for reviewing priority C and D bugs, but in practice we've found it better to make a judgement call for this. The important thing is to remember to review the low-priority bugs eventually. When the list of A bugs dwindles, we review the C bugs, bumping some to priority A, leaving some alone, and downgrading others to priority D. When we start to feel guilty for ignoring the D bugs, we review them, closing some and elevating some to priority C or A.

We also have conducted an audit of our closed PIT bugs. We parceled out our list of closed bugs to individual PIT participants, asking them to evaluate whether the solution was having a positive impact, and to suggest additional followup activities. This was a very productive exercise. We found a few improvements which needed better documentation and better advertising so people would take advantage of them. This is one of the last stages of process improvement—standardizing the process within an organization. We also examined bugs that had been closed with no action taken, and reopened several of them because we hadn't given them enough consideration, or they should have been left open and forwarded to a different Viper module.

Identifying Management Support

Process improvement requires strong management support—this is a well-known fact. But how can you judge whether you have the right level of support? This is somewhat more straightforward in a top-down improvement effort. With top-down support, management takes the initiative in the improvement process. But ours was a grass-roots effort. We went to the test group leader and our manager with the proposal for a process improvement team, and we easily got approval. How could someone say no to improvement? But things got more fuzzy as we discussed the amount of effort we would apply to process improvement. We asked that the whole group be allowed to put 10-20% of their time into process improvement, which would mean that new test development projects we scheduled would be finished later due to the higher overhead. What we got was approval to use any of the group's idle time, without impacting existing schedules, plus endorsement for the PIT leader to spend about 20% of his time in managing the process.

We were not entirely happy with this arrangement, but we knew that the 10-20% figure was arbitrary. We had no supporting data showing how much effort we needed, and what the results would be. So after further negotiations with management, we decided to split our projects into two categories—projects that can be completed within a day, and thus can be squeezed into "idle time", and those that required a larger commitment. For the larger projects, we agreed to write a proposal and present it to management for approval for each project. Once we had a particular project on the table, it was much easier to discuss the effort required and the results we expected. This process worked fairly well. When writing the proposals for large projects, we found ourselves wondering very early in the process how management would respond, and making adjustments in order to increase the odds of gaining approval. Thus, we became more self-managed.

As our merger with Hewlett-Packard grew more distant on the horizon behind us, the test group picked up more aspects of "The HP Way" [PACKARD95]. One of the changes was that we started using MBO—Management by Objectives. Each group member has a specific list of objectives and measurements that management checks against and updates each month.

In many ways, this has been a very useful model to follow—each group member knows exactly what is expected of them, and it is easy to judge whether they have met management's expectations. The

PIT Crew decided that the new test for management support would be to see which PIT projects were explicitly added to the group members' objectives. A number of PIT projects have been completed as part of our individual objectives, meaning that those people working on the projects were judged on their ability to complete the project. When we are evaluated on our participation in a process improvement project, that is a strong indicator of management support.

There have been a few glitches in the MBO approach, though. Once an objective becomes a routine task, we tend to drop it from the objectives and consider it an overhead task. Thus these implicit overhead tasks easily fall by the wayside. This is aggravated by the fact that the objectives tend to be aggressive, sometimes leaving little room during a month to even think about overhead tasks. Also, we have no group-wide metric to track our process improvement accomplishments. This means that there's no process for ensuring that at least a few people have improvement projects as objectives, except when the PIT Crew leader asks at PIT status meetings whether priority A projects are listed in our objectives.

Analysis Paralysis

As part of any process improvement effort, we need to first understand how things currently work. So we kicked off a project to map some of our processes. But the project participants realized that they didn't know how to map processes, so they first set out to learn the process of process mapping. Weeks went by, and the project became known as "Project Death Spiral," for fear that it was in an infinite recursion of learning the process for defining a process to define processes, etc. Finally, the subteam declared that they had mapped out the process for mapping processes, and offered to train others in the technique. We decided to delay the training until just before we decided to do a process map, and to this day, our disdain for "Project Death Spiral" has prevented us from doing this.

PIT Projects

As of March 1998, the PIT database shows 32 projects completed, 33 suggestions closed with no action taken, and 35 suggestions still open. About half of the completed projects involved adding or improving the test group's documentation. This is appropriate, given that you can't improve what you don't understand. Other completed projects include: creating an infrastructure for script language code reuse (perl, expect, shell, etc.), buying sets of books to hand down from one generation of coop students to the next, creating a "task shell" to simplify running tests manually, and sending notification to an internal newsgroup when someone adds a new item to our "Problems & Solutions" web page.

One of the most successful projects was one requested by our customers. Due to the high cost of our lab computers, we needed to improve the performance of our test suites. We added a fairly simple feature that increased our parallelism, and as a result improved performance three-fold in certain scenarios.

We have also had a few episodes of what I call "improvement by osmosis". For example, we had a suggestion to make a typing tutorial available to the test group in order to improve our productivity. Several PIT participants laughed at the idea, not being terribly comfortable discussing personal productivity issues. We gave the bug a low priority, but a few people tried the typing tutorial on their own and made large improvements. So even when PIT takes no direct action, just looking at the suggestions can inspire individuals to make improvements.

We received a lot of good feedback from our coop students and new employees. New employees are only new once, so we try to get their impressions about possible improvements before they become jaded and start to work around problems subconsciously. We also ask coops for feedback just before they return to their next school session.

Evolution

Just as the efforts preceding PIT went through a good deal of evolution, PIT also has evolved continuously. The PIT Crew and its predecessors have survived a corporate merger, several reorganizations in the higher-level management structure, changes in the test group's front-line manager, and a broadening of the test group's scope from the System Software Test Group to the Integration, Test, and Delivery group. The fact that the PIT Crew still exists indicates that PIT has become a core part of the group's culture, and that we have successfully standardized the meta-process of process improvement.

Right from the beginning, we knew enough about Quality processes that we knew we needed to show measurable results. Unfortunately, we have not fulfilled that desire. It is very difficult to measure the result of improving our documentation or making a tool easier to use. So we've justified our existence through testimonials from those who are affected by our results, and by counting the number of completed projects. If we can get more involvement from other parts of the lab, we will have more success with recording relevant metrics.

Another thing we wanted to establish from the beginning is that PIT does not desire to hold a monopoly over all the test group's improvement projects. If someone identifies a necessary improvement and finds the staff to do the work, PIT does not get in the way—there is no need for the PIT Crew in those cases. PIT's real value is taking ownership for “orphaned ideas.” Someone will suggest an improvement, but doesn't have the time to implement it themselves, or maybe it isn't a high priority and they want PIT to prioritize it along with all the other ideas in the database. In these cases, PIT steps in and prioritizes the ideas and makes sure to staff the top few items on the list. However, PIT has gotten a reputation as being a black hole—some bugs in the database languish for months or years without any action taken. We need to do a better job of informing people that any individual is free to work on any PIT bug regardless of the PIT Crew's priorities, as long as they find the free time for it or put it in their objectives. PIT makes no guarantees except that they will work on the bugs that they deem the most important.

I was very surprised by the attendance we had at early PIT status meetings. We met for one hour each week, and averaged at least 75% of the entire test group membership. Perhaps everyone recognized that we had plenty of things to improve! It was several months before participation dropped to about 40 or 50%, and we reduced the meeting frequency to bi-weekly in order to focus more energy on the actual improvement projects and less on management overhead. We would get a core group of participants, and others who happen to be actively working on PIT projects. Now, participation at the meetings is at the 20-30% level, and we have one or two status meetings per month. This is probably a sustainable level of involvement in the improvement process. These numbers do not reflect the work on improvement projects, just the management of the process.

The PIT leadership has also evolved. From the beginning, I did not want to directly lead PIT's efforts. I have been accused of having my head too far in the clouds and asking the group to make changes that are too ambitious (or maybe I just didn't want to repeat the Stuntman affair). So I recruited a leader, and I agreed to be the PIT champion and the facilitator for PIT's meetings. This is a role I continue to fill today. So far, we have rotated through three PIT leaders, and the dual leader/facilitator arrangement has worked very well. The PIT Crew has made a few minor decisions that I personally did not agree with, but in those cases I had to sit on my hands to avoid being a tyrant.

Comparison to Similar Efforts

I observe some interesting contrasts between the PIT Crew and other improvement efforts. One very noticeable difference I've seen concerns a group that doesn't have a formal continuous improvement process at all—the Medical Emergency Response Team (MERT) at our site. At MERT's informal

lunchtime meetings, whether it's on the agenda or not, the subject of their recent performance always comes up, and they always talk about improvements they can make to their process. For example, they once had difficulty flagging down an ambulance after it arrived on our rather sprawling campus. Within days, they had added orange flags to every first aid kit in the building, and MERT put the flags to good use shortly thereafter. They have a passion for improvement that I've never seen anywhere else. This passion is probably related to MERT's mission. An improvement to MERT could affect the quality of someone's life, or make the difference between saving a life and losing one. ITD members probably don't think they are making the same level of contribution to society. Another factor is that the MERT volunteers have limited medical training. They know that during an emergency they are pushing the limits of their knowledge, and that knowledge is only exercised occasionally. So they don't get complacent like a more highly experienced group might.

A "Productivity and Quality" group I observed at another HP site also provides a contrast. They use a top-down approach to quality—improvements are driven by management directive. The P&Q group has a full-time staff, and I imagine that there is little worry about management support. However, when I asked how they dealt with grass-roots improvement suggestions, the answer was that they didn't do this at all. Many of PIT's projects have been small improvements, making day-to-day life at work easier. With a top-down approach, we might not be able to make these small improvements.

I also try to pay attention to improvement efforts in the manufacturing arena. Quality in manufacturing tends to get much more attention than software engineering quality, probably because the process for manufacturing quality control is more established. I participated in a manufacturing defect reduction project, and found that they were drowning in data. They struggled for weeks, trying to decide how to turn the defect data on its ear in order to find a pattern. The situation is quite the opposite for PIT, which has barely any data at all. Many quality control techniques that have been developed for manufacturing processes don't apply directly to a software development process. However, general Quality principles certainly do apply, and in many cases it's only a matter of adaptation to use the same techniques for software.

Full Circle

With a bit of irony, I note that we have come full circle. Through the initiative of another HP lab, we are actively working on the "Universal Test Harness" (UTH), which eventually will replace or run on top of at least half a dozen test harnesses and test tool user interfaces. Rather than dying with a whimper like Stuntman, UTH's first incremental release is now in the testing phase and will soon be released for beta test. The project did have trouble getting approval for the staff that it needed, but a collaboration with ITD gave the project enough critical mass to continue as planned. By deciding to release the tool in incremental chunks, and by listening carefully to his customers, the UTH project leader is very likely to lead a successful project that will have a large impact on productivity. Stuntman lives.

For a long time, we have wanted to broaden PIT's influence. We have sidelined a number of potential projects because they exceed ITD's scope. For example, we have a long-standing bug in the PIT database that calls for a Capability Maturity Model assessment. However, ITD is probably too small for a CMM assessment to provide much benefit. For the organization that contains ITD, however, or perhaps one level higher up, using the CMM makes good sense. We had hoped that other groups at our site would learn by example and start their own process improvement efforts with which we could form partnerships. This would be a good first step toward a broader improvement initiative. But without anyone championing the effort, this has not happened.

There is a change on the horizon. Our lab management is considering creating a full-time productivity group. This would give us the of a larger base of support, and it would open up many possibilities that are currently out of PIT's reach. My hope is that we'll benefit from the steadiness of course of the top-down approach while not abandoning the advantages of a grass-roots approach which gives us grass-

roots buy-in and gives the stakeholders a voice in the improvement process.

I'd like to express my appreciation for the efforts of all the people who are part of this saga, including John Wong, Wallace Davis, Allan Koh, Jeff Bay, Mike Sluyter, Dave Cline, and everyone who helps to keep PIT's spark alive in the future.

References

[CROSBY79] Philip B. Crosby, *Quality is Free: The art of making quality certain*. New York: McGraw-Hill, 1979. ISBN: 0070145121.

[CROSBY96] Philip B. Crosby, *Quality is Still Free: Making quality certain in uncertain times*. New York: McGraw-Hill, 1996. ISBN: 0070145326.

[DEMING86] W. Edwards Deming, *Out of the Crisis*. Cambridge, Mass.: Massachusetts Institute of Technology, Center for Advanced Engineering Study, 1986. ISBN 0911379010.

[ONLINE] See the comp.software.testing FAQ at <http://www.faqs.org/faqs/software-eng/testing-faq/> for links to a wide variety of information about software quality. See the "World Wide Web resources" section for a list of web index sites that will lead you to tons of online information.

[ORGS] Look for local chapters of these organizations or other local organizations like these in your area—

American Society for Quality, <http://www.asq.org/>, 800-248-1946. ASQ Software Division, <http://www.asq.org/about/divtech/softdiv/swqweb.html>.

Dallas-Fort Worth Association for Software Engineering Excellence, <http://LoneStar.rclub.org/ASEE/>. ASEE is a Software Process Improvement Network affiliate, <http://www.sei.cmu.edu/participation/spin/>.

Dallas/Fort Worth UNIX Users Group, <http://www.dfwuug.org/>. DFWUUG is an operating system users group that is not vendor-specific.

InterWorks, <http://www.InterWorks.org/>. InterWorks is an example of a vendor-specific users group.

IEEE Reliability Society, <http://www.ieee.org/society/rs/>.

See the Organizations section of the comp.software.testing FAQ referenced in [ONLINE] for a list of software quality organizations around the world.

[PACKARD95] David Packard, *The HP Way*. New York: HarperCollins Publishers, 1995. ISBN 0-88730-747-7.

[STIMSON98] Carl Stimson, "The Quality Improvement Story," presentation to the Dallas section of the American Society for Quality, February 26, 1998.

[DENT96] Leslie A. Dent, "The Quality Improvement Core Team: A Grass-Roots Approach to Process Improvement." *Software QA Magazine*, Vol. 3 No. 1, 1996. Reprinted from the proceedings of the Pacific Northwest Quality Conference.

[ALEXANDER96] Hilly Alexander and Margie Davis, "Jump-Start Your Stalled Improvement Effort." *Software QA Magazine*, Vol. 3 No. 1, 1996. Reprinted from the proceedings of the Pacific Northwest

Quality Conference.

[TAMMANA98] Prathibha Tammana and Danny Faught, "Software Defect Isolation." Presented at the 13th Annual High Performance Computing Users Group, March 1-5, 1998, Dallas, Texas, and is published in the proceedings of InterWorks '98, April 25-30, 1998, Santa Clara, California.

About the Author

Danny Faught has five years of software testing experience on supercomputer and workstation operating systems. He is currently technical lead of the Integration, Test, and Delivery group in Hewlett-Packard's Richardson System Software Lab. He has been published in *Software QA Magazine* and at the Quality Week '97 conference, and served on the Advisory Board for Quality Week '97 and '98 and on the Program Committee for ASSET '98. He is the maintainer of the Frequently Asked Questions file for the comp.software.testing newsgroup, and co-founder of the swtest-discuss mailing list. Mr. Faught is a member of the American Society for Quality.