In-the-SPIN

Newsletter of the Boston SPIN

Issue 51 January 2003 **Editors: Sheila Lynch, Judi Brodman**

From the Editor

Greetings Fellow SPINners,

I want to speak to you this month about volunteerism. As the snow falls gently in my yard, I am sitting comfortably in the warmth of my house putting the finishing touches on this month's newsletter. As usual, the issue is full of informative articles pertaining, in one way or another, to software process improvement. You'll find January speaker, Johanna Rothman's illuminating article discussing how to measure the effectiveness of bug fixes, Ron Kay's tongue-in-cheek description of how and why he became a Certified Software Quality Engineer and Barb Purchia's humorous comparison of dog training to people correction. There are also summary articles of our recent meetings, including roundtable write-ups. These articles are all written, on a volunteer basis, by members of our Boston SPIN and we are grateful for their contributions. We know that our newsletter has a wide readership, based both on the poll we took at the December meeting and on responses we receive via email from readers all over the world.

But we would like to involve more of you in Boston SPIN volunteer activities. We always need facilitators and scribes for our round table birds-of-a-feather discussions. Every month we are looking for someone to summarize our main speaker's presentation. And if you have a special interest, you could submit a discussion of it for a Feature Article.

Please speak to Judi or me about contributing to In-the-SPIN. We'll be happy to talk with you.

This month, we are also pleased to welcome a new sponsor, William George Associates, a PMI Registered Education Provider (see the last page of this newsletter for more information). Our sponsors allow us to offer high quality programs, not to mention free refreshments, at no cost to you. Without them, it is doubtful we would continue to draw over 100 software professionals each month to our meetings. We have a sponsor brochure if you know of a company that might be interested in becoming a Boston SPIN sponsor.

Before closing, don't forget that January 22 is the last day to register at the early bird rate for the SEPG conference at the Hynes Convention Center in Boston on February 24-27.

Sheila Lynch, Co-editor, *In-the-SPIN*,
email comments to salynch@mitre.org

Please note that for January only, we will meet on the second Tuesday of the month, January 14.

Boston SPIN Software Process Improvement Network

Established January 1993

IN THIS ISSUE . . .

Speaker Spotlight

Using the Fault Feedback Ratio to Predict Project Progress

by Johanna Rothman

Johanna Rothman, a long-time Boston SPINner, is the principal of Rothman Consulting, Arlington, MA.

Most of us track faults (also called defects, problems, issues, bugs) during the system test part of the project. However, many project managers don't track how many of our fixes are successful and how many fixes are bad—either introducing a new defect or not completely fixing the original defect. If you're looking for better scheduling of your system test and project completion, start measuring your Fault Feedback Ratio, the FFR.

Here's how to calculate your fault feedback ratio:

$$\text{Fault Feedback Ratio} = \frac{\text{Fixes that require more work}}{\text{All the fixes}}$$

When I've measured successful projects, the FFR stays under 10% throughout the project, meaning that no more than one in 10 fixes are problematic as the work progresses. (Don't be deceived by a low FFR and a high overall fault count. With a large total count, the defects can be too overwhelming to successfully manage.) A low FFR and a not-too-high overall defect count suggest the developers have a relatively easy time finding and fixing the problems. The testers don't have too much trouble keeping up with testing the fixes, because the fixes haven't broken other pieces of the software. The project team is progressing.

On the other hand, an FFR of 15% or higher means that people are spending significant time finding and fixing problems. A higher FFR and a low defect count may mean that the problematic fixes are in the same part of the code base. A higher FFR and normal or high defect counts, may mean the developers may have trouble detecting where to fix the problems, and the testers may have trouble verifying the fixes are good.

Once the FFR hits 20%, the project team is not making progress. The team is spending too much time re-fixing problems they thought were already fixed and retesting those fixes.

On one project, the FFR started at 18% in the implementation phase, when the project team started to track their defects. Because the developers had completed the design before they started coding, they had trouble fixing the defects quickly. To make up the time, they took shortcuts for fixing the defects, and stopped doing peer reviews on the new code.

By the time the project was supposed to start system test, the FFR was up to 23%. The project had met its previous milestones, but the team was unable to predict the start of system test. Their previous progress was an illusion because uncorrected defects remained in the code. With an FFR of 23%, developers had to take the time to understand each problem and how it affected the rest of the code base, to reduce the fix time for each problem. The project looked stalled, even though the developers were now successfully fixing the problems.

The technical lead suggested a "bug bash," where everyone tries to find and fix problems, but the project manager suggested an alternative: a two-week period where every fix required peer review. At the end of the two-week period, the FFR was down to 6%, a dramatic improvement. The team decided to peer review all fixes from them on, an effective technique for keeping the FFR low. You may not catch all of the problem-fixes using this approach, but you'll catch many of them.

Measure your FFR to see if you're spending too much project effort on fixing fixes. FFR is not linear with effort, so an FFR of 10% does not mean that you spend 10% of your time fixing fixes. You may well be spending more than 20% of your developer time on those 10% of the fixes. In one organization I consulted to, the FFR was 22%, but the developers spent almost 80% of their time fixing fixes.

Track FFR as soon as you start tracking defects, as early in the project as possible. If you start measuring the FFR only at the end of the project, you've missed an opportunity to see how your defects are affecting your project's progress. Calculate the FFR for the entire project each week, not by developer or by code area, making sure the data can't be associated back to a specific person. Your project staff is allowed to make mistakes when they fix problems; your job is to see whether the bad fixes are causing other problems in the project. Use the FFR data at your project team meetings as an early warning sign, to see if your project progress is an illusion or real.

Johanna Rothman observes and consults on managing high technology product development, looking for the leverage points that will make her clients more successful. You can reach her at jr@jrothman.com or by visiting www.jrothman.com.

Feature Article

On Becoming a Certified Software Quality Engineer

By Ron Kay

Ron, an Intel employee, has been the Boston SPIN treasurer since last year. He recently passed the ASQ Certified Software Quality Engineer Exam. This is his story!

You may think I'm exaggerating and I admit I have not taken a comprehensive audit, but I think I now am the only person in my local neighborhood who has been a member of AARP for more than 5 years and who concurrently is a brand new ASQ Certified Software Quality Engineer (CSQE)! Also, soon after I got my certificate this past July, I brought my certificate along with a mere $1 to Boston and was allowed to ride the subway all day! If these two facts don't make you envious, I now can show my certificate at the Super Stop&Shop Deli and buy a (for Boston SPIN meeting) cheese platter for only $5.95/lb!

ASQ and Certification

Seriously though, the American Society for Quality's (ASQ) goal is to be the world's recognized champion and leading authority on quality-related issues. Its Software Division de-

velops the ASQ CSQE program. ASQ certification recognizes individuals who have demonstrated proficiency within a specific area called the Body of Knowledge. Specifically, the CSQE exam was designed for those who understand software quality development, implementation, inspection, testing, verification, and validation; they also can implement development and maintenance processes and methods.

SQE certification is a formal recognition that one has demonstrated proficiency in software quality engineering; it requires education and/or work experience and demonstrated knowledge thru successful completion of a written exam. Hence, it is peer recognition of competence since one has passed a (tough) exam developed by industry subject matter experts covering the most important aspects of software quality engineering.

Why I Decided to Pursue It

I first heard about the ASQ CSQE certification a few years ago at one of John Pustaver's SQA monthly meetings, which was held in Lexington, MA. I read a bibliography of recommended reading material, which concerned me since it was quite broad. However, last winter, Eric Patel spoke about various certification programs during a SQA meeting at Sun in Burlington, MA. I decided to take RapidSQA's review course and I also hoped that my, ahem, extensive software development and quality experience would help pull me thru. Then I could become the only CSQE in my SQA group and possibly be the only one at my Intel, Hudson, MA facility, which has over 2,000 employees. Wow! I've got to order my ASQ CSQE shirt and cap now so I can wear them every day at the office!

Moreover, my new exposure to the CSQE Body of Knowledge has motivated me to continue to read books that are listed in a bibliography that can be found in the book *Handbook of Software Quality Assurance* (see partial list of references at the end of this article).

Have I applied anything that I learned from participating in the CSQE certification process? You bet! For example, the ASQ Code of Ethics is part of the Body of Knowledge. It includes the principal of being honest and impartial, integrity, etc. In the January, 2003 issue of Fast Company magazine (page 97), Patrick Harker, Dean Wharton School, stated that "Truth will be the mantra of business for the next few years." The last several weeks, ethics is the area that I have spent some of my extra time on at work. For the next few years, it could be an important spot where progress of any software organization can be improved!

Why Bother?

Let me just say that I do not feel snubbed in the least for the fact that no announcement of my attaining SQE certification was ever made at any of our daily (10-person) SQA group status meetings, nor at any monthly (50-person) group meeting. Instead, I'm presently waiting and confident that the big announcement will be made at my company's annual meeting!

Anyhow, certification could be an important step in career advancement. It helps to ensure that professional skills are kept current, can provide credibility in a job interview, and could lead to higher pay and faster career growth. Since companies are increasingly seeking highly-qualified candidates, certified candidates may be requested more for interviews and placed on consulting jobs sooner. You may be saying to yourself, yeah, dream on buddy!

More companies have formally recognized ASQ certification as one way to ensure that their workforce is proficient in the principles and practice of quality. Supporting certification also demonstrates a commitment to quality. A 1996 International Data Corp survey of more than 250 IT managers found that most of them believed that certified personnel are worth about $10,000 ($12,000 2003 dollars?) per year over non-certified personnel.

CSQE Qualifications

- The candidate must have 8 years of experience in one or more of the Body of Knowledge topics – 3 of those years must have been in a decision-making technical, professional, or management position. Up to 5 years will be waived if the candidate has a degree from a college, university, or technical school.
- ASQ membership, Professional Engineer registration, or signatures from two ASQ members verifying the candidate's qualifications as a practitioner of quality.
- A successful candidate must pass a 4-hour written 160 multiple-choice question exam.

Preparing for the Big Day

A completed application for "Certification as a Software Quality Engineer" must be mailed about 2 months before the exam date. In the Boston area, (9-week) refresher/review course(s) are available. Such a course is helpful for motivation, so that you are more apt to study and review consistently over a two-month+ time period. Thankfully, the exam is open-book/–notes, but unfortunately, no sample questions and answers are allowed in the exam room.

At work, over a period of a few weeks, I requested multiple times that my company pay for me to register for the course and exam. As the course registration deadline neared, fortunately I became aware of the fact that I could sign up for the course (thru RapidSQA) before actually paying. Then I had more time to get a check cut at my office before the refresher course started. The course was limited to 15 students and I was the last and 16th to get enrolled!

I took RapidSQA's 9-week course this last spring (2002). A similar course will be taught by Eric Patel and Dan Goddu from April 3 to June 4, 2003 (Thurs. nights, 1 Tues.). The course registration deadline currently is February 28, 2003 and the ASQ exam registration deadline is April 4, 2003. For more info, refer to:
http://www.rapidsqa.com/pages/353343/index.htm

ASQ CSQE Body of Knowledge and References

I. General Knowledge, Conduct, and Ethics (16 questions)

Software Quality Engineering (Deutsch & Willis; Prentice Hall) ISBN 0-13-823204-0

II. Software Quality Management (30 questions)

Handbook of Software Quality Assurance **(**Schulmeyer & McManus; Van Nostrand Reinhold) ISBN 0-442-00796-5

III. Software Engineering Processes (26 questions)

Managing the Software Process (Humphrey; Addison-Wesley) ISBN 0-201-18095-2

A Manager's Guide to Software Engineering (Pressman; McGraw-Hill) ISBN 0-07-050820-8

IV. Program and Project Management (24 questions)

Software Engineering, A Practitioner's Approach (Pressman; McGraw-Hill) ISBN 0-07-050783-X

V. Software Metrics, Measurement, and Analytical Methods (24 questions)

Metrics and Models in Software Quality Engineering (Kan; Addison-Wesley) ISBN 0-201-63339-6

VI. Software Verification and Validation (V&V) (24 questions)

Testing Computer Software (Kaner, Falk, Nguyen; Van Nostrand Reinhold) ISBN 0-442-01361-2

VII. Software Configuration Management (16 questions)

Software Configuration Management (Babich; Addison-Wesley) ISBN 0-201-10161-0

Re-Certification

ASQ (CSQE) has a Maintenance of Certification program that requires re-certification every three years. Can't hardly wait!

Committee Spotlight

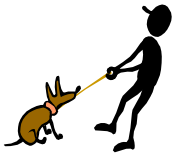
Everything I Needed To Know, I Learned From My Dog.

Submitted by Barb Purchia, Boston SPIN Chair, Rational Software Corporation

So there I was, at the vet with my beagle, Floppy. She was finished with the cursory checkup in the examination room and was now with the vet getting the works (manicure, pedicure, ear cleaning, etc.). I was looking around for anything to read. I'd already checked out all the dog and cat posters in the room and wasn't ready to look at infectious diseases, flea and tick problems, dog and cat anatomy, or animal obesity brochures (which always have a picture of a beagle on the cover).

Something on the bulletin board caught my eye. It was an article by Raymond J. McSoley on *Proper Correction, A Vanishing Art (Doghouse Dogma #3).* Somehow this article struck a chord with me. I knew it was too late for Floppy, but maybe it would work for me! After all, isn't dog training really more about training the owner than training the dog?

With apologies to Mr. McSoley, I have taken his methodology (and most of his article) and converted it from dog correction to people correction. This method is based on the correction versus reaction technique. In the former method, we help a person to learn that a particular behavior is inappropriate, while with the latter; there isn't even the slightest hint of effectiveness. The following looks at what ingredients are necessary in a recipe for effective correction in two parts, short and long term. Short-term goals focus on getting the person's attention. Long-term goals help improve the relationship.



Short-term goals for effective correction:

Help the person understand that the particular behavior is unacceptable. There are many techniques for this. It is preferable not to yank the person's chain when applying this method.

Make sure the person is calmer at the end of the correction than they were before you began. In order for a person to learn, they must focus on you. In order to focus on you, they must be calm. We don't teach the person; we help the person learn. The key here is to make sure that you both have time to focus on each other.

Long-term goals for effective correction:

These require that you have established a relationship based on mutual trust and respect.

To affect correction, the person must understand that he is the follower and you are the leader. If he doesn't understand that, he will not accept correction from you. In this case, you may have more experience in an area or perhaps the person's behavior is causing a problem. Try to be non-judgmental in the discussion. Explain what the cause and effect are. Try not to use terms that could derail the conversation like "You always" or "You never." The person could find one instance when something did or didn't happen and miss the point of what you're trying to accomplish.

Loudness is not firmness. It is the antithesis of firmness. Loudness is merely a vocal expression of our emotional state. It could imply frustration or anger, which may be the case; however, the focus should be on what you're saying not how you're saying it.

Effective correction requires what McSoley calls "quiet firmness." Firmness is an attitude having its roots in simple confidence. We must be confident that our form of correction

will be effective. Sometimes role-playing a tough situation helps to prepare you for expected and unexpected responses.

The correction must, in addition to being effective, allow the person to retain their dignity. You may still have to work or live together.

You will never gain the leader role through correction. You need to be there already.

Anger is always counter-productive, leading to conflict and confrontation, which, over time, corrodes the heart of the relationship.

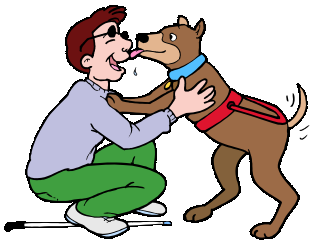
Eye contact and words spoken properly are all necessary to proper correction. Sincerity also helps.

Attempted correction after the behavior has occurred makes us feel better but is never productive. It's best to address it as soon as possible. Otherwise, we may stew over it and it can bother us more each time it happens.

Don't be an "eighty percenter." When you correct someone, you must be 100% focused on the person. If some of your mental focus is not on the issue at hand, like running to a meeting or worrying about what needs to be done, you would be better to forget the correction until you are in the proper mindset. Don't worry; it'll probably happen again.

Multiplicity of corrections – Start over with the short-term goals.

A good person-to-person relationship depends on mutual trust and respect. Proper correction requires a proper mind-set. Feeling guilty about correcting a person negates any possibility of the correction being effective. Part of your responsibility means being able to correct effectively, when necessary.



Learning how to correct is an art and as such requires practice. Understanding the other person's temperament and personality will factor greatly upon how you correct the person.

Lastly, never allow your ego to take charge of correction. The ego never corrects a person properly. It can't; it is always too absorbed in itself. In India, there is a great saying over the doorway in many homes. It reads, "Leave Your Ego with Your Shoes."

Thanks to Raymond J. McSoley and Doghouse Dogma #3, *Proper Correction, A Vanishing Art.*

Thanks also to Floppy for bringing me to the point where I can learn life's lessons from her!

Dear SPIN Doctor

A "GOOD" Requirement –

what does 'good' mean?

by Judi Brodman © 2003

Dear SPINners:

A software developer was recently reading requirements statements to me from a Marketing document they had received. The only thing I heard that resembled a requirements statement was the word "shall". As I listened to the statements being read, I thought of the software developer who would be building 'a product' based on these statements and the tester who would be testing the product against these statements. How would they build from and test against statements that used words such as "adequate", "sufficient" and "make work with"? In no case, was there an interface document, performance specifications, referenced. Even we who think of ourselves as software wizards can't build a satisfactory system based on that kind of ambiguity. (Notice the word "satisfactory" meaning meeting the customer's minimal expectations.)

The very next day I talking with a friend who recalled a conversation where the software developer, when asked how testing had gone, responded that testing went well until the software was given to the user. What does that tell us - that the user wasn't part of defining and reviewing the requirements? Maybe the writer of the requirements (the software developer/marketing person) didn't know how to clearly define what the software would do or didn't know what the user really wanted.

Let's take a second to review the definition of a requirement - a requirement is a statement of the software's behavior, performance, reliability and environment; it is a contractual agreement with your customer that MUST be satisfied by the developer; it is a statement that is testable.

What characteristics do the set of requirements need to have to enable you to build and verify software that satisfies your customer's needs? The IEEE states the following characteristics:

- Complete – having all its parts or elements; thorough, all-inclusive
- Consistent – not self-opposed or self-contradictory; holding firmly together; cohering
- Correct – accurate; conforming to fact or truth
- Testable – capable of being examined (human or machine) for the purpose of evaluating performance and capabilities
- Unambiguous – not open to various interpretations; having only one interpretation

- Free of design – independent of the design; unfettered by design
- Traceable – capable of being tracked or followed (backward and forward); to be able to follow the history, the course, or line of
- Communicable – capable of imparting or transmitting knowledge; usable
- Modifiable – capable of being separated easily; easily changes and updated
- Non-redundant. - not repeated elsewhere

How do you assure that your requirements have these characteristics? Work with your customer to define the requirements. This can be accomplished through the use of brainstorming meetings, storyboards, demos, use cases, interviews, questionnaires – whatever works to clearly define what your customer wants. Conduct group discussions and walkthroughs/peer reviews to ensure that you are "answering the mail". Trace the requirements to your design and code – simply – so you know you are designing/coding what is required – that you are not building the Cadillac when the user wanted the VW. AND, last but not least, take a requirements analysis course coupled with a requirements writing course.

If the requirements work is performed upfront, think of all the trouble and heartache you can save yourself and your customer during testing and turnover.

This column is for you; let's make a difference! Send your comments and questions to "Dear SPIN Doctor" at brodman@LOGOS-Intl.com. Sign them or use a "pen-name" – I respect your confidentiality.

"The SPIN Doctor"

Judi Brodman is CEO and co-founder of LOGOS International, Inc., a software management consulting company focusing on software process improvement.

January Meeting

Featured Speaker – Johanna Rothman

Shattering the Myth of the Inadequate Tester

Why do projects "fail" in the test phase? Do teams have inadequate testers? Are they not doing adequate testing? Is the entire test organization inadequate? We've heard blaming comments about testing and product quality since organizations decided they needed testing groups. Instead of blaming the testers for being inadequate, let's talk about the testing. Why are so many organizations dissatisfied with their testing activities and perceive those activities to be inadequate? Why are we building organizations whose staff can't perform the required work?

In this presentation, Johanna will look at how organizations tend to hire testers and other technical people, and alternatives to those techniques. This talk is for you if you'd like to refine your hiring requirements for great testers (or other great technical staff), or if you'd like to become a great tester (or be hired as one).

About the Speaker:

Johanna Rothman consults on managing high technology product development. She helps managers, teams, and organizations become more effective by focusing on project management, risk management, and people management. Johanna uses pragmatic techniques to help her clients apply effective practices that create successful teams and projects.

A frequent speaker and author on managing high technology product development, Johanna has written numerous articles and is now a columnist for Software Development, StickyMinds.com, and Computerworld.com. Johanna publishes Reflections, an acclaimed quarterly newsletter about managing product development. Johanna is a member of the clinical faculty of The Gordon Institute at Tufts University, a practical management degree program for engineers, and served as the Program Chair for the Software Management Conference for two years. Johanna's book Hiring Technical People will be available in 2003.

November Meeting Synopsis

Featured Speaker – Capers Jones

Software Quality in 2002 – a survey of the state of the art

by Dave Heimann, Boston SPIN Webmaster

On November 19, Caper Jones of Software Productivity Research gave a presentation on "Software Quality in 2002: A Survey of the State of the Art". He based his talk on data collected from about 12000 total projects. The talk began with a list of new lessons for software quality in 2002:

- Businesses are tightly coupled in "supply chains"
- Poor quality in one company can affect scores of companies

- Poor quality drives away clients and loses business
- Poor quality can lead to expensive litigation
- Quality and security are becoming intertwined
- Web-based "content" is a special case (i.e., graphics, sounds)

and continued with the following observations:

1. In terms of delivered defects per function point, quality ranges from 0.13 for best-in-class to 0.75 for U.S. average for quality to 3.05 for poor quality (malpractice).
2. Techniques such as Formal Inspections, Joint Application Design and Quality Metrics using Function Points are good, with more than a 90% success rate. Those such as Total Quality Management, Independent Verification & Validation are mixed, with around a 50% success rate. Those such as ISO 9000, Informal Testing, and Manual Testing are poor, with a less than 25% success rate.
3. The larger the project, the lower the proportion of defects from coding and the higher proportion from management/support, defect removal, and paperwork. Also, the larger the project the higher the defect rate per function point (defects are proportional to function points raised to the 1.15 power) and the lower the defect removal efficiency.
4. In terms of delivered defects per function point in 2002, information software and Web software have more than average, while system, commercial, military, outsourced, and embedded software have less than average. The SEI CMM level is very significant, with each higher CMM level decreasing the defects per function point by multiples of at least 3.
5. Of ten industry-wide defect causes, the ones requiring the most effort to fix the defects are, in order, requirements problems, design problems, interface problems between modules, and logic/branching/structural problems.

The talk discussed the effect of implementing various combinations of Formal Design Inspections, Formal Code Inspections, Formal Testing, and Formal Quality Assurance. While the best effect was obtained by implementing the four techniques in the above order, the worst outcome of using two of the techniques was better than the best outcome of using one of the techniques, with similar results for using three methods vs. two and all four vs. three.

An interesting aspect of the talk covered what claims each side in software litigation made. Plaintiffs often asserted schedule overruns, cost overruns, poor quality, and false claims, while defendants asserted requirements changes, new demands by clients, being rushed by clients, and client refusal to cooperate. Both sides cited problems with ambiguous clauses, informal cost estimates, no formal quality estimates at all, no use of formal inspections, inadequate milestone tracking, personal disputes, and too-late independent audits.

All in all, Capers had a lot of insight and opinions, backed up by data, into the current world of software quality.

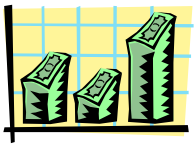
Software Process Improvement Roundtable

ROI – How to Measure Improvements

Facilitated by Judi Brodman, CEO LOGOS International, Inc.

Scribe: Dave Hallowell, Managing Partner, Six Sigma Advantage, Inc

This month's Software Process Improvement (SPI) Round Table discussed different "Returns" expected from investments made in improving the software development process. The group also discussed ways in which these returns/improvements could be measured.



The group started the discussion by deciding what returns would be beneficial to their organizations:

1. **Increased Productivity** related to our work-product output in size, for a given staffing level and project duration,
2. **Reduced defect insertion** rates and better **defect containment** rates,
3. **Meet milestones,**
4. **More Reusability,**
5. **Better Maintainability,**
6. **Increased Market share** / better penetration into the market,
7. **Increased Return (repeat) business**, and
8. **Decreased level of rework.**

When we discussed how you could obtain those returns, i.e., what investments we could make to obtain those returns, the following ideas were put forth:

1. For reduced defect insertion/better containment
    a) **Better understood requirements** (not necessarily more/bigger requirements…could be more succinct, smaller volume of data)
    b) Better **documentation** flowing down throughout the life cycle
    c) Spending more time **designing** – not getting hurried to code
    d) Using **prototypes** /evolutionary/**incremental design** to clarify requirements and design concepts. Noting that during those iterative cycles the re-

visions of designs and plans are **not 'rework'** as they are an intended part of the process, done when such changes are low cost and low risk.

e) **Peer reviews.**

2. To Meet Milestones
   a) More detailed **work breakdown structure** – granular enough that planning and tracking are really do-able; save the WBS for lessons learned on new projects.
   b) **Smaller tasks** for better tracking of schedule
   c) Better **estimation** (of effort, duration, defect levels)
   d) Better **matching of skills** to the work
   e) **More training** to meet needs
   f) More **management discipline** (reducing interruptions and changes in goals/scope that disrupt the original plans)
   g) Better **tracking** (at the granular levels mentioned above) using tools and techniques.
3. For better **Reusability**
   a) **Produce customizable, general purpose architecture** that supports and facilitates new uses,
   b) **Object-oriented thinking** and tools at the front end; not necessarily tied to an OO language. The thought process yields benefits in many implementation domains.
   c) Make sure you **design for reuse in the beginning**.
4. For better **Maintainability**
   a) Procedures/functions with **few interfaces**, simple, short construction (small modules),
   b) Lots of **comments; more documentation**
   c) Discipline regarding **documentation changes** to match all code and other changes,
   d) **Change management / configuration management tools** to automate and enforce good practices.

December Meeting Synopsis

Speaker– Unmesh Gundewar

Staff-less SQA: An Effective SQA Alternative

by Jim Withall, Boston SPIN Membership Chair

Biography:

Unmesh Gundewar is a Consultant for EMC's Software organization and chairs a company-wide process improvement effort. He works with senior executives to establish standard practices for software development and project management, focusing on key business goals to improve time-to-market, predictability, product quality, and employee satisfaction.

He holds a patent on the ProjectModel, "A Simple Approach for Achieving CMM Level 2". He has published numerous papers and is a frequent speaker at local PMI (Project Management Institute), SPIN chapters, and at PMI and SEI conferences.

Gundewar holds a master degree from the University of Florida and he is PMP certified.

Presentation:

Gundewar opened his presentation by stating the key to corporate success and survival is closely linked to the timely delivery of products and services with superior quality. To improve this quality, many corporations are suing CMM (Capability Maturity Model) for process improvement. According to the August 2002, Process Maturity Profile, published by SEI (Software Engineering Institute) "Software Quality Assurance is the least frequently satisfied Level 2 Key Process Area."

An in depth definition of the CMM Level 2 KPA (Key Process Area), Quality Assurance, was presented. Software Quality Assurance Goals include:

1) People following standards,
2) The standards communicated, and
3) Issues that cannot be resolved within the project are escalated.

The typical organization establishes a "SQA" group, and makes them responsible for SQA compliance. SQA groups often run into a number of issues. The people doing SQA work don't like checking on the work of others, morale within the unit is often low (the 'I don't get any respect' syn-

drome), there are often funding issues, and there are problems with having to police and enforce standards. All of these factors can contribute to low compliance in this critical SQA area.

EMC has taken an alternative strategic approach, called "Staffless SQA". Instead of a separate SQA group they have made "product quality" part of the overall responsibility of the Program Manager. The role of the Program Manager includes "develops cross-functional program plans, drives program to completion, and ensures that the product is delivered on time and meets the EMC quality goals." The SQA responsibility has been transferred to project teams who provide management with direct project visibility. Since senior management is part of the project review function, the approach eliminates the need to escalate non-compliance issues to management.

Gundewar provided a review of EMC's Product Lifecycle which included:

- EMC's Software Development Process activities and checkpoints,
- EMC's Leadership Meetings Workflow Escalation Process, and
- Developing and displaying compliance metrics.

The Software Quality Assurance Key Process Area was also reviewed:

1) The Program Manager owns the SQA function.
2) Adequate resources and funding are provided to ensure that checkpoint reviews are planned and conducted.
3) Individuals are trained to perform quality assurance activities.
4) The results of various review activities are periodically reported to affected groups.
5) Problems and defects detected are documented, tracked, and addressed.

In conclusion, Program Managers are the key part of the successful implementation of "Staffless SQA" The Program Manager "does quality" as part of their overall responsibility.

A copy of this presentation is available on the Boston SPIN web site, http://www.cs.uml.edu/Boston-SPIN/bos-SPIN.html.

SPIN Information

The Boston SPIN is a forum for the free and open exchange of software process improvement experiences and ideas. Meetings are usually held on third Tuesdays, September - June. Boston SPIN welcomes volunteers and sponsors. There is no charge to attend the meetings. Additional information about the Boston SPIN can be found at our Web home page: http://www.cs.uml.edu/Boston-SPIN/bos-SPIN.html.

For more information about our programs and events contact Barry Mirrer, Program Chair, bmirrer@alum.mit.edu.

Cancellations (including weather)

Starting at 3pm, we'll notify you via email to the SPIN distribution list, we'll post the notice on the SPIN web page, and we'll send the cancellation announcement to Channel 7 TV and WRKO AM 680.

SPIN Meeting Location

Boston SPIN meetings are held at The MITRE Corporation in Bedford.

Please be aware that MITRE has advised us that, due to increased security concerns, you will need a Picture ID for admission to the SPIN meetings. We encourage you to leave all carrying bags, backpacks, and briefcases behind (i.e., in your car). Otherwise, you should be prepared to have these opened and inspected upon arrival.

MITRE's campus is located at 202 Burlington Road (Route 62), Bedford. Directions can be found on our Web site: http://www.cs.uml.edu/Boston-SPIN/bos-SPIN.html or on The MITRE Corporation Web site.

Sponsors

The following organizations/individuals support the Boston SPIN:

- The MITRE Corporation, http://www.mitre.org/
- Raytheon Company, http://www.raytheon.com/
- Edelman & Associates, http://www.edeltech.com
- UMASS – Lowell, http://www.cs.uml.edu/

And please **welcome Our New Sponsor!**

- William George Associates, www.williamgeorgeassociates.com

William George Associates will be one of our sponsors for the January, February and March meetings. They are a PMI Registered Education Provider, who delivers project management training for professionals pursuing PMP(r) certification, and provide consulting services aimed at improving the ROI of projects across the enterprise.

They're also offering a special $150 discount off the training class fee to Boston SPIN members. Check out their web site.

If your organization is interested in sponsoring Boston SPIN, please contact SPIN Steering Committee member Rick Brenner at rbrenner@chacocanyon.com or ask any Steering Committee member for a brochure.

Email Lists

To receive Boston SPIN specific notices, send an email to:

- Jim Withall, Boston SPIN membership chair, at withall@rcn.com.

Future Programs

We welcome your suggestions for future Boston SPIN programs. We are always looking for interesting speakers. If you'd like to speak at Boston SPIN, please review the criteria specified on the Boston SPIN Web site before sending an abstract to:

- Barb Purchia, Boston SPIN chair, at bpurchia@rational.com.

Newsletter Call for Articles

The *In-the-SPIN Newsletter* is always in need of new and interesting articles dealing with process improvement, software development methodologies, project management and other related subjects that may be of interest to our readership. Please send general correspondence or articles that you would like to have considered for publication to:

- Sheila Lynch, Co-editor of In-the-SPIN, at salynch@mitre.org
- Judi Brodman, Co-editor of In-the-SPIN at brodman@logos-intl.com

Back issues of the *In-the-SPIN* Newsletter can be found on the Boston SPIN Web site: http://www.cs.uml.edu/Boston-SPIN/bos-SPIN.html.

Upcoming Meetings

Our remaining 2002—2003 schedule follows.

DATE	SPEAKER	TOPIC
Jan. 14, '03	Johanna Rothman	The Inadequate Tester
Feb. 18, '03	Sam Guckenheimer	Agile Testing
Feb. 24-27, 03	SEPG Conference	Boston
Mar. 18, '03	TBA	
Apr. 30, '03 (note date)	Jim Highsmith	Agile Software Development
May 20, '03	Linda Northrup, SEI	
June 17, '03	Robin Goldsmith	Non-CMM Software Quality Improvement