

In-the-SPIN

Newsletter of the **BostonSPIN**

Issue 44 January 2002

Software Process Improvement Network

Editors: Judi Brodman
Sheila Lynch

Editorial

Last month we were again visited by one of the greats in the software field - Watts Humphrey. His presentation, *What Is Excellence? Process Improvement for Individuals, Teams, and Organizations*, which traced the software process improvement movement from its beginnings in the mid-eighties to today, is well summarized by Barbara Purchia.

In this issue we also have two terrific articles dealing with scheduling:

- In Committee Spotlight, Johanna Rothman discusses why staff members agree to a schedule they know they can't make in hopes of beating the last person to finish, and how the Project Manager can end this 'schedule chicken', and
- The Featured Article is Part I of a two part article dealing with the small actions that shred our schedules, in some cases, without our even realizing it. These actions 'leak time out of any schedule' says Richard Brenner.

In addition, we are again fortunate to be able to feature a Speaker Article, this month by January's SPIN presenter, James Bach, on Exploratory Testing. James defines exploratory testing and discusses how it can be used in conjunction with our scripted testing.

The SPIN Doctor also discusses her current experience as part of a successful CBA-IPI assessment team and what the positive results of the assessment mean for the organization.

In each issue, we are attempting to put together relevant information in the form of articles, columns, and recaps from the previous month's meeting and Roundtables in hopes of providing our SPIN members with worthwhile information to support their current interests and management/technical positions. We hope we are hitting our goal of providing you, our readers, with worthwhile information – write us and let us know!



Judi Brodman, Co-editor, *In-the-SPIN*, email comments to brodman@logos-intl.com

Letter from the Chair

The end of year causes me to pause and reflect on the year behind us and what the possibilities are for the year ahead. I just wanted to review with you what we do for you, our members:

Speakers: Tom DeMarco, Ed Yourdon, Watts Humphrey, Capers Jones to name a few.

Roundtables: focused groups or "birds-of-a-feather" discussions, with a facilitator, to stimulate and moderate discussion including the

Book Club: where you can discuss books related to software development, process, engineering, and project management; these groups are held during the Networking portion of the SPIN meeting.

SIG: a newly formed special interest group centered on testing and proving to be a popular part of our programs.

Newsletter: in depth coverage of happening at SPIN, speaker articles, summaries of meetings, editorials and much more.

New Boston SPIN Web Site (www.bostonspin.org): easier navigation, new features including the **Discussion Board** - a place to post jobs, events, rides, resumes.

Do It Yourself Resource Table: copies of Job Postings, Announcements, resumes, Call for Papers and other Notices.

And the cost? Just remember - all these programs are free as a result of continued sponsorship by our patrons.

And what you can do for SPIN? Volunteer! Please contact volunteer@bostonspin.org to find out what YOU can do for SPIN.

Happy and Safe New Year!! Linda McInnis, Chairperson

BostonSPIN Established January 1993
Software Process Improvement Network

IN THIS ISSUE . . .

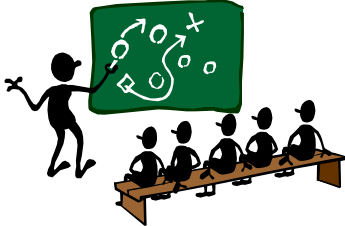
Editorial.....	1
Letter from the Chair.....	1
Committee Spotlight: Ending Chicken Schedule	2
Speaker Spotlight: What is Exploratory Testing.....	2
Feature Article: Schedule Shredding, Part I	3
Dear SPIN Doctor.....	5
December Meeting Synopsis	6
January Meeting Preview.....	9
Upcoming Meetings	10
Announcements	10
Boston SPIN info.....	10

Committee Spotlight



This issue's Committee Spotlight article is contributed by Johanna Rothman, president of Rothman Consulting. © 2001

Ending Schedule Chicken



Imagine this: You're in a project team meeting, listening to the project manager reviewing the major milestones. Your PM says, "We're on target to make the release date, right?" Everyone seems to be nodding, or somehow

agreeing with the project manager, but every single person in that meeting knows he or she will not meet this schedule. All you have to do is be ready before the last person who claims to be on time actually finishes. Schedule Chicken has struck again.

Schedule Chicken is a universal phenomenon. It's very hard to speak up in a room of your peers, or worse, your managers, and say "Um, my part's not going to make it on time." You look like the only one who can't keep a schedule. Even if your management doesn't blame you for being late, you might feel as if you've let the group or the project down.

Schedule Chicken starts when everyone is afraid to be honest and hopes someone else will blink first. An astute project manager recognizes the need to test the project information she receives at project team meetings. One technique to end schedule chicken is to get anonymous information.

Step 1: Discover how people feel about their date commitments.

Divide a flip chart into probabilities ranging from 0% to 100% in 10% increments across the bottom. Along the side, put the numbers ranging from 1 to the number of people in the meeting. Label this chart the "Confidence I Will Make My Dates." Make a similar chart labeled "Confidence I Have in the Overall Schedule."

Ask everyone to fill out a sticky note with a numeric value of their confidence in their dates, where 0% is no confidence, and 100% is absolute confidence they will make their schedule dates. Collect those notes, allowing each person their anonymity.

Ask everyone to fill out a separate sticky with the numeric value of the confidence they have in the overall schedule. Collect those notes, again retaining anonymity.

Sort each set of sticky notes by number. Tally the number of notes by percentage for each chart. See how many people are at which level of confidence, both for their own dates and for the overall project. Look to see where the levels of confidence are not very high.

Step 2: Use this information to discover what's really happening on the project.

If people are not feeling high confidence in their dates, you will not make the schedule. Find out why people aren't feeling confident in their dates. Then, decide if it's time to start solving problems in the project. Make sure you're not preventing people from giving you bad news.

If people are feeling good about their dates, and NOT good about the overall schedule, you have a different problem. Start examining why people don't trust their peers' dates.

If people are feeling good about their dates, and good about the overall schedule, then maybe things are fine for now.

On one project, the project manager did this, and wasn't sure what the data was saying. There were five people at the meeting, and each person had a different probability, (0%, 10%, 40%, 80%, 100%) they would make their dates, non-clustering data. When you have non-clustering data like this, look for unequal amounts of work, or a difference of opinion about what the milestones mean. The percentages for the overall schedule confidence were 0%, 10%, 20%, 80%, 80%. In this case, some of the developers assumed that as long as they finished *their* work by the time the product had to ship, they were going to make the project dates.

The testers and the writers knew they couldn't make their dates if the developers took the entire schedule to finish their work. The project manager used the original anonymous data to say, "What's happening here?" The PM was able to use the data to draw out the different groups' assumptions. The discussion helped the developers see they had low probabilities of making their date too. Then, the entire project team redefined their milestones and what they could deliver by the desired ship date.

Schedule Chicken is a common problem on software projects. As a project manager, you need to discover what's going on for the people on your project, and solve some smaller problems before they lead to schedule crises.

Speaker Spotlight



This article first appeared as a column on <http://www.StickyMinds.com> and is reprinted with the author's permission.

What is Exploratory Testing? And How it Differs from Scripted Testing

by James Bach



Exploratory software testing is a powerful and fun approach to testing. In some situations, it can be orders of magnitude more productive than scripted testing. I haven't found a tester yet who didn't, at least unconsciously, perform exploratory testing at one time or another. Yet few of us study this approach, and it doesn't get much respect in our field. It's high time we stop the denial, and publicly recognize

the exploratory approach for what it is: scientific thinking in real-time. Friends, that's a good thing.

Concurrent Test Design and Execution

The plainest definition of exploratory testing is test design and test execution at the same time. This is the opposite of scripted testing (predefined test procedures, whether manual or automated). Exploratory tests, unlike scripted tests, are not defined in advance and carried out precisely according to plan. This may sound like a straightforward distinction, but in practice it's murky. That's because "defined" is a spectrum. Even an otherwise elaborately defined test procedure will leave many interesting details (such as how quickly to type on the keyboard, or what kinds of behavior to recognize as a failure) to the discretion of the tester. Likewise, even a free-form exploratory test session will involve tacit constraints or mandates about what parts of the product to test, or what strategies to use. A good exploratory tester will write down test ideas and use them in later test cycles. Such notes sometimes look a lot like test scripts, even if they aren't. Exploratory testing is sometimes confused with "ad hoc" testing. Ad hoc testing normally refers to a process of improvised, impromptu bug searching. By definition, anyone can do ad hoc testing. The term "exploratory testing"--coined by Cem Kaner, in *Testing Computer Software*-- refers to a sophisticated, thoughtful approach to ad hoc testing. In the last decade, James Whittaker (at Florida Tech), Cem Kaner and I have worked to identify the skills and techniques of excellent exploratory testing. For one example of a fully defined and articulated process of exploratory testing, see the *General Functionality and Stability Test Procedure for Microsoft's Windows 2000 Compatibility Certification program*. This document is publicly available on Microsoft's web site, or on my site at <http://www.satisfice.com/tools/procedure.pdf>.

Balancing Exploratory Testing With Scripted Testing

To the extent that the next test we do is influenced by the result of the last test we did, we are doing exploratory testing. We become more exploratory when we can't tell what tests should be run, in advance of the test cycle, or when we haven't yet had the opportunity to create those tests. If we are running scripted tests, and new information comes to light that suggests a better test strategy, we may switch to an exploratory mode (as in the case of discovering a new failure that requires investigation). Conversely, we take a more scripted approach when there is little uncertainty about how we want to test, new tests are relatively unimportant, the need for efficiency and reliability in executing those tests is worth the effort of scripting, and when we are prepared to pay the cost of documenting and maintaining tests.

The results of exploratory testing aren't necessarily radically different than those of scripted testing, and the two approaches to testing are fully compatible. Companies such as Nortel and Microsoft commonly use both approaches on the same project. Still there are many important differences between the two approaches.

Why Do Exploratory Testing?

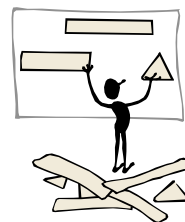
Recurring themes in the management of an effective exploratory test cycle are tester, test strategy, test reporting and test mission. The scripted approach to testing attempts to mechanize the test process by taking test ideas out of a test designer's head and putting them on paper. There's a lot of value in that way of testing. But exploratory testers take the view that writing down test scripts and following them tends to disrupt the intellectual processes that make testers able to find important problems quickly. The more we can make testing intellectually rich and fluid, the more likely we will hit upon the right tests at the right time. That's where the power of exploratory testing comes in: the richness of this process is only limited by the breadth and depth of our imagination and our emerging insights into the nature of the product under test. In the rapid testing classes at Satisfice, Inc., we have equipment that watches testers invent tests in real-time. When the instructor makes a new suggestion for what to test, or provides new information to the testers about the product, we observe and measure how a roomful of exploratory testers reacts to that information. Free from the encumbrance of pre-documentation, they immediately incorporate new ideas into their tests.

Scripting has its place. I can imagine testing situations where efficiency and repeatability are so important that we should script or automate them. For example, in the case where a test platform is only intermittently available, such as a client-server project where there are only a few configured servers available and they must be shared by testing and development. The logistics of such a situation may dictate that we script tests carefully in advance to get the most out of every second of limited test execution time. Exploratory testing is especially useful in complex testing situations, when little is known about the product, or as part of preparing a set of scripted tests. The basic rule is this: exploratory testing is called for any time the next test you should perform is not obvious, or when you want to go beyond the obvious. In my experience, that's most of the time.

Feature Article

Schedule Shredding: It's the Little Things We Do Part I

This issue's feature article is Part I of a two part article contributed by Richard Brenner, a consultant and principal of Chaco Canyon Consulting (www.ChacoCanyon.com) and a Boston SPIN board member. Copyright © 2001



If you've ever pondered the deep truths of project management, you've probably wondered why projects are so often so late. If everyone working on a project—and certainly everyone who isn't—wants the project to be done on time, why can't we finish on time?

We turn first to the usual explanations—aggressive schedules, inaccurate estimates, and unanticipated disasters. But in the end, a big part of the problem is in the little things we do.

Many of our actions make local sense, but global nonsense. That is, we make decisions which do achieve our immediate local or parochial goals, but which have serious, negative consequences in other parts of the organization. We might be unaware of these consequences, while the people who suffer those consequences might be unaware of their cause. Nevertheless, the decisions and their consequences are real, and they shred schedules.

In this issue and the next, I present a collection of practices that leak time out of any schedule. This month, I'll emphasize the personal level and next month, the organizational level

Image polishing

We all want to make a good impression, but is it really worth waiting for the color printer to free up? And do we really need beautiful PowerPoint, when a bulleted list on a sheet of paper will do?

I don't know of any measurements or estimates of the cost of the frills we use in the everyday presentations that we give to each other. It would be difficult to measure, because the people who are spending their time on presentation frills would probably find out that we were measuring it, and then they would cut back. I do have a sense of how much time I've spent on such things personally, either self-directed or otherwise, and I look back on it in amazement.

How do we get to a place where the project is three months late and still it seems to make sense to spend 20 minutes fiddling with a slide set's color scheme? How do we miss this obvious inversion of priorities?

Two sets of players contribute to this cultural quirk: the presenters and the audience. The audience, typically management, responds to well-crafted presentation graphics. Audience members tend to confuse form and content, and they telegraph their confusion to presenters. Meanwhile, presenters try to use any technique they can to make the audience more receptive. These two groups are bound to find themselves in a spiraling escalation of presentation craftsmanship. Project delays and excessive use of expensive consumables (color printer supplies) are the result of this arms race of presentations.

To control the escalation, create a new cultural norm in your organization: all presentations must be black-and-white and free of animation, video or audio, unless the content demands it.

Adversarial gatekeeping

In many organizations, when people who work on projects submit forms such as requisitions for equipment or personnel, the forms are processed by administrative staff who serve an important gatekeeping role. Their function, in part, is to ensure that the right people spend

organizational resources for the right things in the right way. Often these gatekeepers feel no particular urgency relative to the projects that are underway in the organization. Rather, they focus on validating the forms and processing them accurately and with dispatch. This emphasis is usually built into the design of the job, and it can directly conflict with project objectives.

For example, in one purchasing organization, a series of complaints about slow processing and high processing costs led managers to reorient purchasing administrators. They decided that to save time, administrators would no longer correct problems, even minor problems, in requisitions. Instead, they would merely flag problems and return the requisitions to the initiator. This practice led to several expensive and irritating project delays. Engineering managers responded by allocating a group of their own administrators to the task of reviewing and correcting requisitions prior to submitting them to Purchasing for processing. In effect, they chose to duplicate the purchasing administrator function, in a way that served the needs of projects, by providing rapid, customer-oriented assistance with requisitions. They justified the high cost of this duplication because it eliminated expensive project delays. Still, at the organizational level, it was a waste.

Generally, gatekeeping activity is not seen as adding value. Pressed to reduce their own costs, and to push for ever-higher throughput, gatekeepers can become adversaries of their internal customers, as they adopt tactics that often lead to project delays. Organizational leaders must find ways to bring gatekeepers and project teams together. Both groups add value, and both groups must learn to respect each other.

Interruptions

When the phone rings, when someone drops in, when we're paged, or when the computer advises us that we have mail, we're interrupted. And we're interrupted when we're working and we overhear a particularly juicy hallway conversation, or when we hear a building-wide page or announcement.

Interruptions are expensive. By some estimates, we require about 15 minutes to refocus our attention completely following an interruption. Conservative estimates suggest that we lose 25% of our time to interruptions.

We can reduce interruptions without loss to anyone. For example, most of us get so much email that there's no need to have our computers announce at the appointed interval that the inbox has something in it—we know it always will. Just turn the darn thing off. In fact, go do that right now. I'll be here when you get back.

Next time you're thinking about phoning someone, consider email instead. Unless you urgently need to converse right now, or unless you know that your recipient doesn't read email, email can be more effective, when we consider the cost of interruptions.

If there's someone in your work life who insists on phoning you, consider setting your phone to route calls to voice mail temporarily. You'll be rewarded with an interruption-free chunk of time, and you might actually get something done. Check your messages at intervals that work for you. Tell your family to always call you on your cell phone.

Drop-in interruptions are more difficult to manage. They're best dealt with at the organizational level, by declaring quiet hours, in which drop-in visits and phone calls are discouraged.

Similarly, facility-wide pages and announcements are also organizational issues. To make a case against facility-wide pages, consider that in a facility with 1000 professionals, a single page costs 1.4 person-months of lost productivity. At the rate of one facility-wide page per month, you're spending \$100 per person per year on paging. Why not buy personal paging services for everyone who is paged more often than once a month? The investment pays for itself in time not lost by everyone else.

Changing course

When we change our minds about the goals of a project, delays often result. Changing goals can cause delays even when the changes *narrow* the scope of the project. Often we're tempted to make major changes because they seem smaller than they really are.

Imagine that you're an office tower developer, and that your 188-story building in Singapore has in place about 80 stories of steel, 60 stories of concrete floors, and 40 stories of glass skin. One thing that won't be on the agenda of a status review meeting is switching to a different steel alloy for floors 1 through 50.

No one would consider changing something so basic so late in the project. Yet, in product development in other industries—especially software—this sort of thing happens maddeningly often. When schedules slip and budgets overrun, our first instinct, too often, is to change the design.

Using computers for new product development is one source of this problem. Whether the product is software, integrated circuits, or even legislation, products developed with software tools might never exist physically until development is fairly advanced, if ever. When we're building a skyscraper, the physical form of the building itself helps us see the folly of many proposed changes, but products developed using software tools often lack physical form. So we feel free to move the goal posts.

Address this problem by creating and maintaining a list of decisions that are set in concrete and can't be changed. Keeping these items off the table frees up additional time for discussing things the team can actually do.

In the next seven days, collect your own organization's personal schedule-shredders. Which ones are your own personal favorites? What can you do to manage them? If you're concerned that changing just your own behavior might not make a visible difference, pass this article around. If you work together you *can* make a difference.

Dear SPIN Doctor

Who, What, Why, and How – a CBA-IPI Assessment

by Judi Brodman © 2002



Dear SPINners:

I've just completed a CBA-IPI assessment and those of you who have been through this exercise know what that means – exhaustion. This CBA-IPI was performed on three projects within a small organization (staff of 25 people). This was the organization's first CBA-IPI, and at the end, it was rated a solid Level 2 against the SEI's Capability Maturity Model (SW-CMM) AND satisfied many of

the goals at Level 3!!

First, let's acknowledge that achieving Level 2+ is not a trivial accomplishment for any size organization but is particularly difficult for a small organization. Why? Because there is a limited number of staff members to perform both project and software process improvement (SPI) activities. This staff limitation causes some hard decisions to be made by Senior Management and this constant struggle for staff between project and SPI activities can easily cause a small organization to abandon SPI in the pursuit of contract work (and dollars). This was not the case in this small organization – Senior Management provided SPI support consistently and made the hard decisions.

To understand this accomplishment, I need to provide you with a little background information: what a CBA-IPI assessment is and how this small organization achieved Level 2+ and how long it took them. Then, let's look to the future and see what achieving Level 2 means for them and what their future plans are.

CBA-IPI Assessment

A CBA-IPI (Capability Maturity Model Based Appraisal for Internal Process Improvement) is a method for assessing an organization's software development capability using the SW-CMM as a reference model. The minimum requirements for a CBA-IPI are:

- the assessment must be led by an authorized Lead Assessor,
- team members must have taken the Intro to CMM course,
- interviews must be conducted with project leads and staff,

- data collected through interviews and document review must be corroborated and validated,
- draft findings must be reviewed by participants, and
- final findings must be presented to the organizational sponsor and also returned to the SEI.

How and how long

In order to understand how this organization accomplished Level 2, you should know a bit about its environment. They have one project that is very well defined, long term, and is producing the same type of product for their customer every month or so. The other two projects involved developing new software, in one case, web-based,. In the two software development projects, requirements were not known early on in the projects. They had to develop a process where they could work with their customers to develop the requirements - a spiral life cycle with RAD/JAD sessions with their customer. They planned and estimated the activities leading to each RAD/JAD session. Once the requirements become more solid, they used the Delphi estimation method to estimate the work ahead. They also had piloted and brought on board tools that supported their Configuration Management and Project Management processes as well as providing a repository for all their process and project artifacts. They developed a process to be used by all projects and supported this process with procedures for activities that they wanted to be repeatable from project to project. They documented Project Plans, followed them, updated them, and, in the end, were successful both with their customers and with satisfying the SM-CMM.

They started their improvement program in 1998 with a mini assessment. Following that assessment, they had a number of delays due to lack of funding but, in the fall of 2000, they started in earnest to work towards their Level 2 goal.

The results of achieving Level 2

With that said, what does achieving Level 2 + mean to this small organization. Currently, the organization is now the ‘poster child’ for process improvement within its own company. Other organizations are already calling on them to lend SPI expertise. Outside their own company, Level 2+ means that this organization will be taken seriously when bidding on new work with old and new customers – government or commercial.

Future plans

The organization has made the commitment to achieve Level 3 in the next year!

At the June SPIN meeting we will be the feature speakers and will be discussing how this organization achieved Level 2+ and what it learned along the way. The presentation will be given in two parts – the senior manager of the organization will present lessons learned from this experience and I will present what techniques I used to help this organization achieve Level 2+. We think this

presentation will provide valuable SPI information to other small (and large) organizations.

May you all have a happy, healthy, prosperous 2002!!

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"

December Meeting Synopsis

The following synopsis is contributed by Barbara Purchia, Boston SPIN Vice Chair, Director of Engineering Operations, Rational Software

What Is Excellence? Process Improvement for Individuals, Teams, and Organizations Speaker: Watts S. Humphrey



Watt Humphrey, the world renowned author and “father” of the Software Capability Maturity Model, spoke to a full capacity audience in December about excellence and process improvement for individuals, teams, and organizations.

Watts talked about how the Software Capability Maturity Model (CMM) was developed while he was at the Software Engineering Institute (SEI). A study group of the SEI, MITRE, and Air Force had been asked to determine why major system contracts were over budget and late, with an average delay of 75%. They produced a list of questions that would help define what a good contractor did. Watts then applied the Phil Crosby Quality College structure for evaluating organizations to the questions and this became the framework for the CMM.

For this presentation, excellence for organizations means that the organization consistently delivers quality products, on committed schedules and for planned costs. The main premise of his presentation was that “organizational excellence” must involve teams and engineers. “Team excellence” means that the environment is orderly and well-managed and must include the engineer doing excellent work. “Engineering excellence” means delivering quality products, on committed schedules and for planned costs. Each area (organization, team, and engineer) must follow a similar process of defining their process, planning for each job, recording and tracking performance. The organization is aided by the CMM. To provide help with the latter two areas, Watts developed both the Personal Software Process and the Team Software Process.

Watts discovered that another key to excellence and success is executive support. A defined process, training, and a supportive environment is not enough. Management excellence is also needed to provide the needed training,

support, and oversight, and to make excellent work normal and natural. By focusing all areas of an organization on excellence, Watts identified several major improvement areas, a significant decrease in the deviation from effort and schedule and a dramatic reduction in the number of defects found in acceptance testing and post-release as well as in system test duration.

Organizations, teams, engineers and executives are all needed to help define and build an excellent environment.

December Roundtables

Roundtable #1: Striking the Balance between Scope, Schedule, Staffing and Quality

Facilitator and Scribe: Johanna Rothman



How do you really make tradeoffs between cost, duration, resources and quality? Many of us learned project management as the techniques of balancing a three-legged stool: cost, schedule, and quality. However, as software people, we're quite aware that the people we staff on the project have as much to do with the project's quality and success as cost and schedule. Not only is the staff part of the project success equation, but also the working environment and the feature set you choose also have effects on the project's chance of success. In this roundtable, we discussed what project managers really trade off, to make their projects successful.

Issues discussed:

Power: Who's in power in the organization (for this project)? Does the project have a sponsor? Whose product is it? The people in power tend to make the tradeoff decisions.

CMM level: Instead of power, there may be a methodology to make decisions.

What phase is the project in?

How much trouble is the project in? If project is in a lifeboat situation, organizational power has much less influence.

What happens when managers don't understand the staffing requirements: which people when, how many? Some managers think that understaffing leads to increased productivity, but we've never seen that.

If you want to make tradeoffs, you have to know what you're trading off. Requirements and planning are critical for the project. Even in a short time, you can use RAD to help with quick requirements definition and project planning.

We've met people who "bluffed" about requirements (Yes, we have to have this) during the requirements phase, but were willing to negotiate about requirements at the end of project (Just ship it anyway). Some of us wanted more experience with how to ask what's important.

We then discussed whether it was all politics, that the politics of a project were parallel to the issues of requirements. The political connection can drive who works on the project, and

what they do. One way to deal with politics is to ask: What will this feature generate in revenue?

If you have too many people on the project, take a break and re-evaluate the project and the people skills. With too few people, do risk management.

When making tradeoff decisions, who are your umpires?

Do you have "acceptable levels of defects" for your application (we agreed this depended on the application).

Using release criteria can help, even if they change over the course of the project.

Roundtable # 2: Boston SPIN Test SIG (Special Interest Group)

Leaders/Hosts: Linda Blizzard and Patrick Hughes

The Software Testing Forum is a new SIG in Boston SPIN. Its focus is on software testing *as one of the ways of improving the quality of software*. Its focus is not on quality overall, as quality concerns occur throughout software phases and specializations. We plan on alternating group discussions and presentations by industry experts. "Forum" is defined by Random House dictionary as: "An outlet for discussion of matters of interest to a given group."

On Dec. 18, 2001, two Ajilon employees presented their opinions on some of the more important best practices in test automation. Linda Blizzard is an Ajilon consultant specializing in test automation. Patrick Hughes is a V.P. of Ajilon. Ajilon is an IT consulting company which had purchased Software Quality Associates.

Test Automation Best Practices

The conclusions presented here are a summary from notes taken during the forum, and do not attempt to represent a formal position taken by Ajilon.

The first thing to do before automating tests is to determine if there is a payoff for automating. It is important to compute the break-even point. The break-even point can be computed by comparing the time needed to develop automation, versus the time saved by automation.

Regression testing is one use of test automation where the break-even point should be computed. Two uses of test automation have virtually no other way of testing effectively, so test automation is even more likely to be expected for them. These are load testing and performance testing. Other than these, tests that are to be done only once should not be automated.

Some of the principal disadvantages of test automation are time and money spent on automation development, and a need for more employee skills in the area of test development.

If there is a payoff for test automation, a tool must be selected. The most important selection criteria is object recognition. That is, can the tool recognize objects? Also, does the tool give objects sensible names? Strong functionality in object recognition prevents having to test using X/Y coordinates.

There should be coordination with other teams. Especially, it is beneficial for developers to give unique names to all windows.

Test cases should be written before beginning automation. It is advisable for test cases to be written with upcoming automation in mind, so the test cases support automation.

Use of a tool's functionality to record and capture tests is usually used to initiate development. But, don't expect record/capture to suffice. The code for the test cases needs to be changed so that it pulls data from a spreadsheet or comma-delimited file. This has several benefits, including facilitating the changing of dates. Tests should be grouped so they read from 1 data set.

The forum's leaders are Peter Malpass (Pgmalpass@hotmail.com) and Paul Piper (paulpiper@mediaone.net). Please contact them with suggestions for discussion and presentations.

Roundtable # 3: The Personal Software Process – A Few Unexpected Lessons

Facilitator and Scribe: J. Erik Hemdal

Have you tried applying PSP?

Of the people at the Roundtable, 5 knew about the PSP; two had tried it; one tried it “partway”.

Overall Results

The PSP process “works”, and class exercises demonstrate improved performance, but PSP doesn't scale well. No one reported successfully transitioning PSP to project work.

Using PSP as a “guerrilla” tactic to circumvent the difficulties of an organizational change ultimately did not work out.

There can be a lot of resistance to the detailed time and defect measurement. In one situation, a team used poker chips to keep track of time-on-task. By using chips to track 15-minute blocks of time, the team got fairly good time resolution without a lot of overhead.

One PSP-trained engineer originally resisted the process of defect tracking, design and code review, until he completed an exercise that compiled cleanly on the first try and yielded zero defects in unit test. When the time he planned for testing and debugging ended up as extraneous, he was convinced of the value of reviews.

What it means to be a “good programmer” changes under the PSP. Sometimes a more “junior” programmer ends up doing very well when the results are carefully measured; while more experienced engineers struggle with the new methods. There's a culture shift that needs to be recognized and catered for.

What do you see as roadblocks to instituting PSP?

Organizational pressures

When only 30% of your time is spent on coding, managers want to know what's wrong. Often manager's expectations don't match the PSP reality.

Reward systems

In some cases, if you aren't spending long hours in test and debug, the assumption is that your code wasn't very demanding, or that you aren't “committed” to the success of

the project. So what is rewarded by managers sometimes needs to be adjusted.

In other situations, there might be more support. One participant stated that at Lockheed programmers were chased out at 4:30. It was assumed that you were competent enough to meet your objectives without a lot of overtime.

“Public goods”

We discussed the notion of PSP providing a “public good”. Software without defects is desirable, but there's no way to “pay for it” in most organizations, it's like having clean air or clean water in society. Because of the way we generally organize work, it's difficult to align the team with the goals of PSP. As one example, if a team is measured against achieving a specific schedule date, it's difficult to ask them to add reviews to the process when this might cause a short-term disruption to the plan. And if the objective of the team is simply to “get code to QA”, then there is little incentive to do extra work that improves the quality of that code.

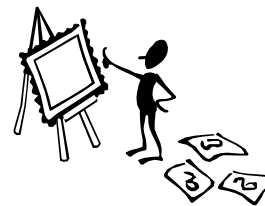
So, just as with the “public goods” of clean air and water, we need a way to demonstrate and quantify the value of the “goodness” that no one has specific incentive to pay for.

We also need to consider what is visible and measurable to the organization outside of a software team. When you are maintaining code, if you add 10% more code and get twice as many defects, you have an obvious problem. When programmers refuse to modify code because it is very fragile, you know something is wrong. You can use PSP to create robust code that can be extended with far fewer problems.

Roundtable # 4: Process Definition

Facilitator and Scribe: David Heimann

1. What is the structure of a process definition? There are four parts:



o Components ("What")

- The items used to produce software, e.g., software development model, templates, standards, requirements, specifications, designs, test plans,

test cases, technical documentation, release notes, databases, spreadsheets, version controls, configuration management, etc.

o Policies and procedures ("How") - The goals for the use of the components and the rules or guidelines for using the components.

o Entry and exit criteria ("When") - The prerequisites for entering a given stage of the software development process (entry criteria) and the requirements for exiting that stage (exit criteria), as well as the criteria for completing the software product development as a whole.

o Measures ("How well") - The quantitative values and metrics used to determine how well the process is being followed and the quality of the software product.

As we use the initial process, we eventually realize which parts are the most important for our particular organization

and focus on them. In addition, we tailor the process to fit the given project and the given group carrying it out.

2. What considerations do we take into account in forming the process (i.e., the "meta-process")? There are several considerations, including:

- o What do we want to accomplish? Before the project starts, write a "success story".
- o How do we best use the varied skills of the software group?
- o How can we have average-skilled programmers perform well?
- o We have an ongoing evaluation and adaptation process.

January 15 Meeting

Roundtable Programs 6:30* - 7:00 PM

Roundtables are focused group or "birds-of-a-feather" discussions, with a facilitator to stimulate and moderate discussion. Please join us for a lively series of discussions during the networking portion of the SPIN meeting, before the speaker. Select the topic of your choice, but come early. The facilitators will determine the number of participants, and "first come, first served."

Roundtable # 1. Creative Ways to Implement Risk Management in Software Development

No time to make a plan for Risk Management?

- * Don't know enough about Risk Management?
- * Can't afford a tool for Risk Management?
- * Your project is too simple for Risk Management?

Roundtable # 2. How to Reduce Schedule or Cost Overruns

Recommended discussion: - Define basic minimum situation in which schedule can be reduced, ask "what's your experience?" What to try, when it works and doesn't.

Roundtable # 3. Boston SPIN Test Forum

Leader/Host: Pete Malpass

This month we'll have a round robin dialogue about each person's experience with test and, in particular, test automation has been. Please come prepared to describe the software type, extent of reuse and test plan, then what type of automation or tools you've used and assessment of the benefits vs. disadvantages.

* **Please Note: This SIG roundtable only will BEGIN PROMPTLY at 6:00 PM.**

January Boston SPIN Book Discussion

Corps Business :

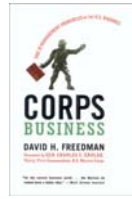
The 30 Management Principles of the U.S.

Marines

by David H. Freedman

Editorial Review: Amazon.com

In *Corps Business*, author David Freedman examines the organization and culture of the United States Marine Corps and sees "the best management training program in America."



For this book Freedman, a senior editor at *Forbes ASAP* and author of *Brainmakers*, trained with the Corps and interviewed scores of marines of every rank to discover 31 management principles "built

around simple truths about

human nature and the uncertainties of dynamic environments.... The Marines are used to facing entrenched enemies, short time-frames, chaotic conflicts, and unfavorable terrain --all of which have come to be hallmarks of the New Economy." Some of the ideas that Freedman encountered include Principle No. 1: "Aim for the 70-percent solution. It's better to decide quickly on an imperfect plan than to roll out a perfect plan when it's too late"; Principle No. 13: "Manage by end state and intent. Tell people what needs to be accomplished and why, and leave the details to them"; and Principle No. 21: "Establish a core identity. Everyone in the organization should feel they're performing an aspect of the same job." It's hard to argue with two centuries of battlefield success, and the wisdom and time-tested management philosophy dissected here should be a valuable prescriptive for any organization hell-bent on winning. --*Scott Harrison*

Come join us on January 15 for a lively discussion of this book.

January Main Program

Writing Testing Lessons?

Speaker: James Bach

Abstract:

A year ago, Cem Kaner, Bret Pettichord and I became frustrated with the preachy and hidebound way that testing textbooks are written. We decided to try to write a different kind of testing book one where we offered, as simply and plainly as we could, useful advice about testing. We managed to write the book in about 5 weeks of furious work. Along the way, we learned a lot about putting wisdom into words; and what we learned might be relevant to any tester, in any company, who is trying to codify a test methodology and pass along what they know.

James Bach is the founder of Satisfice, Inc., a test training and consulting company in Virginia. James specializes in training testers in the art of high accountability exploratory testing. With Cem Kaner and Bret Pettichord, he is the author of *Lessons Learned in Software Testing: A Context-Driven Approach*.

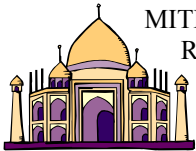
Upcoming Meetings

1/15/02	James Bach	Writing Test Lessons
2/19/02	Linda McInnis	Managing Virtual Teams
3/19/02	Donna Johnson	
4/16/02		Joint ASQ/SPIN Meeting
5/21/02	Tom Lister	To Be Announced
6/18/02	Judi Brodman and Steve Hannigan	Achieving CMM Level 2 and Beyond SPIN Elections

New Meeting Location

Boston SPIN meetings for the 2001-02 year will be 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. SPIN meetings are held in the 'S' building. Directions can be found on our Web site: <http://www.bostonspin.org>

Announcements

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.

Future Programs

We welcome your suggestions for future Boston SPIN programs. Program suggestion forms can be found on our website <http://www.bostonspin.org>. 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 Linda McInnis, Boston_SPIN@yahoo.com.

Sponsors:

The MITRE Corporation
Raytheon Company
Edelman & Associates
Quality Search
UMASS – Lowell (provides support)

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. For more information about our programs and events contact:

Linda McInnis, Chairperson
Boston_SPIN@yahoo.com

Send letters-to-the-editor, and general correspondence to:

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

To receive notification of new *In-the-SPIN* issues and Boston SPIN specific notices, send email to:

withall@mediaone.net

Back issues of the *In-the-SPIN* Newsletter and other information about Boston SPIN can be found at our WEB HOME PAGE:

<http://www.bostonspin.org>

