In-the-SPIN

Newsletter of the Boston SPIN

Issue 50 November 2002

Editors: Sheila Lynch
Judi Brodman

From the Editor

Greetings Fellow SPINners,

Our year has gotten off to a successful start with our first two prominent speakers, Michael Mah and Larry Constantine. This month we welcome back Capers Jones who will discuss the state of the art in software quality.

How fortunate we are to have such a strong interest in software process improvement in the Boston area! We have had over 100 attendees at each of our first two meetings. And, not only do our roundtables continue to draw a strong attendance each month, but you have asked us to extend them to a full hour. Our meetings also provide a wonderful way of networking with our peers, something those of us affected by 'reductions in force' are well aware of.

The Steering Committee continues to welcome your ideas for how we can improve our organization. To that end, a poll soliciting your ideas for this newsletter will be available at this month's meeting. Please take a few minutes to let us know what you think.

And, as always, think about volunteering to help Boston SPIN. We always are looking for new faces to contribute to our organization. New faces bring new ideas and experiences – something that keeps an organization robust

One area where we are in need of help is in facilitating our roundtables and in writing articles about them for In-the-SPIN. Another area we would welcome your help is in obtaining sponsors. If you know an employer or organization that is a potential Boston SPIN sponsor, please see our chair, Barb Purchia.

Now, on another note, the following story, original source unknown, has been making the rounds on the Internet. As I read it, I saw several parallels to our sometimes-frustrating efforts to achieve software process improvement. Think about how you can apply this to your job…

"Start with a cage containing five monkeys. Inside the cage, hang a banana on a string and place a set of stairs under it. Before long, a monkey will go to the stairs and start to climb towards the banana. As soon as he touches the stairs, spray all of the other monkeys with cold water. After a while, another monkey makes an attempt with the same result – all the other monkeys are sprayed with cold water. Pretty soon, when another monkey tries to climb the stairs, the other monkeys will try to prevent it.

Now, put away the cold water. Remove one monkey from the cage and replace it with a new one. The new monkey sees the banana and wants to climb the stairs. To his surprise and horror, all of the other monkeys attack him. After another attempt and attack, he knows that if he tries to climb the stairs, he will be assaulted.

Next, remove another of the original five monkeys and replace it with a new one. The newcomer goes to the stairs and is attacked. The previous newcomer takes part in the punishment with enthusiasm! Likewise, replace a third original monkey with a new one, then a fourth, then the fifth. Every time the newest monkey takes to the stairs, he is attacked. Most of the monkeys that are beating him have no idea why they were not permitted to climb the stairs or why they are participating in the beating of the newest monkey.

After replacing all the original monkeys, none of the remaining monkeys have ever been sprayed with cold water. Nevertheless, no monkey ever again approaches the stairs to try for the banana. Why not? Because as far as they know that's the way it's always been done round here. And that, my friends, is how company policies are made. As Mark Twain penned: "It's not what you don't know that's so dangerous. It's the things you think you know that just ain't so."

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

Boston SPIN
Software Process Improvement Network
Established January 1993

IN THIS ISSUE . . .

Please note: The November Boston SPIN meeting will be held in The MITRE Corporation M Building, instead of the K Building. Directions are posted on the Boston SPIN web site, http://www.cs.uml.edu/Boston-SPIN/bos-SPIN.html.

Speaker Spotlight

The Role of Software Quality in Process Improvement

by Capers Jones

Capers Jones is the Chief Scientist Emeritus, Software Productivity Research, Inc., (an Artemis Company), Burlington, Massachusetts.

Introduction

Developing large software systems has long been recognized as one of the most hazardous business activities of the computer era. Many large software systems are either cancelled, or experience schedule delays in excess of one calendar year and cost overruns that can exceed 100%.

The author and his colleagues at Software Productivity Research have examined the results of about 12,000 software projects between 1983 and 2002 (Jones 2000). Table 1 shows the percentages of projects that are on time, early, late, or cancelled for six size plateaus each an order of magnitude apart:

Table 1: Software Project Outcomes By Size of Project

Probability of Selected Outcomes

	Early	On-Time	Delayed	Canceled	Sum
1 FP	14.68%	83.16%	1.92%	0.25%	100.00%
10 FP	11.08%	81.25%	5.67%	2.00%	100.00%
100 FP	6.06%	74.77%	11.83%	7.33%	100.00%
1000 FP	1.24%	60.76%	17.67%	20.33%	100.00%
10000 FP	0.14%	28.03%	23.83%	48.00%	100.00%
100000 FP	0.00%	13.67%	21.33%	65.00%	100.00%
Average	5.53%	56.94%	13.71%	23.82%	100.00%

Table 1 uses function point metrics for expressing software size. (For additional information on function point metrics, it is useful to visit the web site of the non-profit International Function Point Users Group (IFPUG). The web address is http://www.ifpug.org.) For specifics on using function points and other metrics for quality measurements, refer to Kan 2002.

As can easily be seen from table 1 small software projects are usually successful, but large systems are not. Why not? The main reason for the failure of large software projects is poor quality. The phrase "poor quality" in this context has two meanings:

1. Excessive numbers of defects or bugs
2. Inadequate defect removal activities

In order to know what volume of defects might be "excessive" and what level of defect removal is "inadequate" it is useful to know current U.S. averages. Table 2 shows defects originating in requirements, designs, code, user documents, and "bad fixes" or secondary defects. Table 2 shows the average volumes of defects found on software projects, and the percentage of defects removed prior to delivery to customers:

Table 2: Defect Removal Efficiency By Origin of Defects Circa 2002 (Data Expressed in Terms of Defects per Function Point)

Defect Origins	Defect Potentials	Removal Efficiency	Delivered Defects
Requirements	1.00	77%	0.23
Design	1.25	85%	0.19
Coding	1.75	95%	0.09
Document	0.60	80%	0.12
Bad Fixes	0.40	70%	0.12
Total	*5.00*	*85%*	*0.75*

Table 2 illustrates two unfortunate aspects of average software projects: 1) Large volumes of defects likely to occur; 2) Defect removal efficiency is not very good.

However, when examining the results of software projects developed by leading companies that have implemented successful process improvement programs, it can be seen that total defect volumes are lower than average, while defect removal efficiency levels are better than average. Table 3 illustrates typical results for defect potentials and defect removal levels based on the capability maturity model (CMM) developed by the Software Engineering Institute (SEI):

Table 3: Software Quality and the SEI Capability Maturity Model (CMM) - for Projects of 5000 Function Points in Size

SEI CMM Level	Defect Potential per Function Point	Defect Removal efficiency	Delivered Defects per Function Point
CMM 1	5.50	73.00%	1.49
CMM 2	4.00	90.00%	0.40
CMM 3	3.00	95.00%	0.15
CMM 4	2.50	97.00%	0.08
CMM 5	2.25	98.00%	0.05

As can be seen, levels 3, 4, and 5 are significantly better than U.S. averages in terms of both overall volumes of defects and defect removal efficiency levels. One of the main benefits of achieving the higher CMM levels is better quality control, which pays off in more predictable project outcomes. This raises interesting questions as to exactly what kinds of process improvements benefit defect volumes and defect removal efficiency.

Reducing Defect Volumes
Since the number of defects found in requirements and designs outnumber coding defects, leading companies and leading projects are very thorough in gathering requirements and in producing specifications. Of course reducing coding defects and "bad fixes" are important too.

Some of the methods noted that reduce requirements and design defects include:

- A joint client/development change control board or designated domain experts
- Use of formal requirements and design inspections
- Use of joint application design (JAD) to minimize downstream changes
- Use of formal prototypes to minimize downstream changes
- Formal review of all change requests
- Revised cost and schedule estimates for all changes > 10 function points
- Prioritization of change requests in terms of business impact
- Formal assignment of change requests to specific releases
- Use of automated change control tools with cross-reference capabilities

One interesting aspect of controlling requirements is a reduction in unplanned changes or "requirements creep." Ordinary U.S. projects average about 2% per month in new and changing requirements. Leading projects where the requirements are carefully gathered and analyzed average only a fraction of 1% per month in unplanned changes.

Raising Defect Removal Efficiency Levels
Most forms of testing are less than 30% efficient in finding bugs or defects. However, formal design and code inspections are more than 65% efficient in finding bugs or defects. Therefore all leading projects in leading companies utilize both formal inspections and formal testing. This is the only known way of achieving cumulative defect removal levels higher than 95%. Table 4 illustrates the measured ranges of defect removal efficiency levels for a variety of reviews, inspections, and test stages:

Table 4: Software Defect Removal Efficiency Ranges

Defect Removal Activity	Ranges of Defect Removal Efficiency
Formal design inspections	45% to 85%
Formal code inspections	45% to 85%
Unit test	15% to 50%
New function test	20% to 35%
Regression test	15% to 30%
Integration test	25% to 40%
Performance test	20% to 40%
System test	25% to 55%
Acceptance test (1 client)	25% to 35%
Low-volume Beta test (< 10 clients)	25% to 40%

The low defect removal efficiency levels of most forms of testing explain why the best projects do not rely upon testing alone. The best projects utilize formal design and code inspections first, and then a multi-stage testing sequence afterwards. This combination of inspections followed by testing leads to the shortest overall development schedules, and lowers the probabilities of project failures.

Summary and Conclusions
The phrase "software process improvement" is somewhat ambiguous. The phrase by itself does not indicate what needs to be improved. However from analysis of large numbers of projects that were either failures or quite successful, it is obvious that quality control is the top-ranked issue that needs to be improved. With state of the art quality control, successful projects become the norm. With inadequate defect prevention and defect removal, cancelled projects and disasters are the norm.

References and Readings
Jones, Capers; Software Assessments, Benchmarks, and Best Practices; Addison Wesley Longman, Boston, MA; 2000.

Kan, Stephen H.; Metrics and Models in Software Quality Engineering; Addison Wesley Longman, Boston, MA 2002.

Committee Spotlight

Holey Grails

By Rick Brenner

Rick is Principal of Chaco Canyon Consulting, and an At-Large member of the Boston SPIN Steering Committee. You can subscribe to his free weekly email newsletter, Point Lookout, at www.chacocanyon.com/pointlookout/.

Stepping out of the conference room for a solo break, Ellie closed the door behind her. Another one-hour meeting was gradually turning into an all-day affair, and she was determined not to let it mess up her entire day. She would at least check her voicemail. She did that, and then stopped by Marketing's coffee machine for a refill.

For some reason, Marketing really did have the best coffee. Returning to the conference room, she slid silently through the door and back to her seat. It was like a time warp in there—she had missed nothing. Greg was talking again. Or maybe still talking. He finished with, "The best way to sort this out is to look at the no-cost options first. Then if none of them look OK, we can talk about Denton's idea."

Even though Greg wants to optimize the group's search for a decision, he might actually be introducing an obstacle. His point is that the procedure he advocates is "best." The obstacle arises because most of the problems groups wrestle with have no "best" solution. And even if there were a best, groups rarely address the basic question: "best with respect to what measure?"

Too often, we assume that "best" is knowable—that there is one best way. The assumption permeates our conversation and our thinking. It leads us to trouble, too, because usually we can't define "best." But the real tragedy is that most often, "best" doesn't even exist. Most problems have multiple solutions, each with strengths and weaknesses. What's best depends on your goals and values, and "better" is just as much a trap as "best."

When you notice a group focusing on a discussion of "better" and "best," ask yourself if there is agreement on how to measure it. Without such agreement, call a halt—you're wasting time. Instead, try to forge an agreement on the meaning of "better" or "best," or choose a solution some other way.

Here are some key words and phrases that people use when the discussion is focused on "better" or "best."

Better, best, optimal, optimize, maximal, maximize, more or most effective.

These are the words that often signify absence of a consensus metric. What does "effective" mean, anyway?

Worse, worst, suboptimal, inferior, minimal, minimize, less or least effective.

These are their negative cousins.

We can save a lot of time (or money or energy or trouble or...) if we...

This presumes that saving these resources is a primary goal. Greg was doing this in the scenario above.

If we could remove from meetings any discussion about "better" and "best," unless it's solidly based on a consensus about how to measure "better" and "best," we could all go home a lot earlier every day. Compared to what we now do, maybe that would be better. Or maybe not.

Dear SPIN Doctor

Software Process Return on Investment

Dear SPINners:

Return on Investment (ROI) for Software Process improvements still remains a hot topic. Participants in the SPI Forum (Roundtable) have chosen ROI as the topic for this month's Roundtable and for that reason I am reprinting the ROI column I wrote in 1995. It still covers the information that people need to consider when they are putting together a metrics program and want to collect ROI data to present to their management and peers.

The following article appeared in the June/July 1995 edition of the In-the-SPIN Newsletter and is reprinted with the acknowledgement of the author.

Dear SPINners: Summer Solstice Greetings! I apologize for not being here last month. I probably missed writing the column more than you missed reading it. The weeks right before the SEPG National Conference were very hectic for all of us on the planning committee! I hope that many of you had a chance to attend the conference. Being somewhat biased, I think we produced one of the best conferences thus far for gathering practical experience data... at least that's what you told us!

To open the second day of the conference, there was the infamous "Beantown Brawl" between Bill Curtis and James Bach, exquisitely executed by Herb Krasner. It was, as touted, a very interesting exchange of ideas; and, in some cases Bill and James' views were not so far apart (a warm-up debate for Pres. Bill and Newt?). The "round breakers" were an interesting touch - the final one being Watts Humphrey who managed to add a few comments of his own as he crossed the stage carrying the final round number! We need to create more vehicles similar to the "brawl" for the exchange of differing views; this mechanism allows us to listen and decide for ourselves which side we belong to or which views fit our current situation. I personally think it was a tie... and I'm looking forward to the rematch! How about it Bill and James?

For a first time at an SEPG Conference, there was a SPIN booth in the Exhibit area... and, for those of you who purchased the now famous Dragon T-shirt, "May the Process be with you"! The Boston SPIN booth ended up being a great place to meet other SPINners and I did manage to meet a lot of you there.

In the last column, because so many of you had questions about establishing and continuing meaningful metric programs, I said that I would spend the next few columns discussing metrics and measurement. This month, based on questions and comments that you had following the Return-on-Investment (ROI) presentation that Donna and I made at the conference, I have decided to discuss two areas together -- metrics and ROI.

You all appear to be very interested in ROI data. I assume that the interest stems from the need to present ROI information to your management to ensure that they initiate or continue your process improvement funding. In the early stages of your process improvement program, your management might be content with seeing what other organizations have been able to realize as returns on their investments in process improvement. But at some point, your management will want to see a real return based on their investment -- a real return being a representation of process improvement for your organization based on your organization's investment. And how do you represent the improvement within your own organization? The question of how to calculate ROI within an organization is probably one of the most difficult and critical questions that many of you face right now.

Let me try to quickly and simply present a number of steps that may aid you in answering the ROI question from an improvement point of view, not a dollar point of view, for your organization:

- Define your return(s). This is the hardest definition to agree upon within your organization. For your corporation, a return will be dependent on corporate and customer goals; for your software organization and your project, a return will be dependent on the organization's and project's goals for products, e.g., quality, productivity, etc.; for your business area or program managers, a return may be judged by customer satisfaction and by return or increased business.
- Define the representation mechanism for the return(s). This is the point at which you very carefully choose the metric(s) that will represent the return(s) to all concerned parties.
- Create a "footprint." In order to calculate ROI, you need to have a "footprint", as Dr. Howard Rubin refers to it. The footprint represents where you are now in the areas that you have chosen to track as the representation of your improvement.
- Collect the return data. This data is collected by means of your chosen metric(s).
- Calculate your ROI. A simple ROI is a calculation based on the comparison of your footprint or last measurement with a current or updated measurement.

Even the simple way sounds too simple -- five steps and you have ROI data. But each step above contains areas where incorrect or misguided decisions can send you off in the wrong direction, or worse yet, can cause the demise of your metrics program.

Metrics used for ROI are part of your basic set of metrics. You should be careful not to collect too many or too few metrics.

- Collection of too much data can cause frustration on the part of your software engineers. First, it takes time from their "software duties"; second, perhaps more importantly, the data is never used.
- Too little data can lead to incorrect decisions or conclusions. In the metrics and ROI research that Donna and I conducted, we found that Level 1 organizations were collecting, on average, 11 metrics -- with "collecting" being the key word -- not "using"!

Level 5 organizations were found to be collecting AND using 31 + metrics. Level 1 organizations were not always collecting useful data or the type of data that would be useful in a historical database as they matured. So you also need to be aware of what you will need for historical data as your organization matures.

Therefore, choose your metrics carefully -- make them useful to YOUR organization, and choose ones that will be useful to you at Level 5 as well as at Level 2. Don't be overly ambitious in your choices! And, plan for success!!

If you need to calculate ROI from a dollar point of view, your task will be considerably more difficult. You will have to go through similar steps for investments as I outlined above for returns. Investments can also come in many 'flavors' -- resources (including $$), training, tools, facilities, etc. You need to be able to convert these investments into a dollar value and then convert your return(s), which can be more than just the simple metric representation outlined above, into dollars also. You can then calculate a true ROI. It is not an easy task, nor one to be taken lightly if you want a correct representation of what your organization is accomplishing. And remember, a return may be negative at any time based on the types of investments being made, such as training and tool investments. But the return should take a positive turn in the future if all is working correctly in the organization.

ROI calculation is really too large a subject to cover in this column but I hope that the ideas I have suggested will help many of you initiate your ROI program as part of your metrics program. If you continue to have questions in this area, we can discuss it again. It is a favorite topic of mine.... Can you tell?

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

November Meeting

Featured Speaker – Capers Jones

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

Capers Jones is chief scientist emeritus and executive vice president of Artemis Management Systems. Mr. Jones is an international consultant on software management topics, a speaker, seminar leader and author of 12 books and more than 250 journal articles on software management. He is the founder and former chairman of Software Productivity Research. Before founding SPR, he served as assistant director of Programming Technology at the ITT Programming Technology Center, Stratford, CT. Prior to ITT, he worked at IBM for 12 years.

He began his software career as a programmer in the Office of the Surgeon General in Washington, DC, then became a programmer/analyst for Crane Company in Chicago. Mr. Jones is the designer of several software cost and quality estimation tools, including SPQR/20™, the first commercial software estimating tool to use function points as the basis for sizing source code. He also designed the algorithms for Checkpoint (RM), an integrated measurement and estimation tool.

Roundtables

- Process Improvement forum: ROI - How to Measure Improvements, facilitated by Judi Brodman

- Software Testing Forum: What Low Cost Alternatives Are Available for Automatic Testing?, facilitated by Richard Powers
- Roundtable: Criteria for Successful Configuration/Release Management
- Roundtable: Peer Reviews and Inspections – Proven Best Practices
- Roundtable: Communications Breakdown
- Book club: "Software Runaways (Great Disasters of software," by Robert Glass

September Meeting Synopsis

Featured Speaker - Michael Mah

Software Estimation and Negotiation, "Changing the Game" in a Down Economy

by Sheila Lynch, In-the-SPIN editor

Boston SPIN kicked off their tenth year by welcoming back Michael Mah, who gave a well-received talk on estimation and negotiation for software projects. Michael started by making some observations on current trends in our industry. He pointed out that since the initial Standish Group 'Chaos' Report, that software projects now encounter fewer project cancellations but more overruns. Our productivity is down, our schedules are creeping up and projects are requiring more effort. Some contributing factors to this trend are Internet speed deadline pressures, the effects of the recession on our organizations and the complexity of the applications we build. There is thus more need than ever to effectively estimate and negotiate software projects.

Typically, the negotiations process for estimating software projects becomes adversarial. Michael advocates arguing on the merits of the options rather than taking an 'us vs. them' attitude. We should generate multiple options when preparing estimates, such as best case vs. worst case, use of COTS vs. internal development, more staff vs. less staff, extend schedule vs. compress schedule, etc. The merits of each option can be determined UP FRONT, allowing preparation of an informed estimate based on negotiation of options.

Michael emphasized that most of us have not been trained in the negotiations process. He defined seven important elements we need to consider:

- Communications – We need to develop good two-way communications with our customer. We should explain our reasoning and our process and inquire into theirs. Listen to the customer and try to understand their concerns and constrains. Good communications also require a mature software management process.
- Relationship – Raise issues early. We should resolve to work together to solve problems and always separate the people from the problem. Speak for yourself, not for the customer.
- Interests – We can negotiate more easily by clarifying our interests and concerns, by asking why, by iterating our understanding of the customer's needs.
- Options – Options can be invented by looking at the software equation, Project size = (Process productivity) x (time) x (effort). Work together with the customer to make trade-offs on schedule, functionality, cost and reliability. Do a risk analysis.
- Legitimacy – Legitimacy means seeking an outcome that is "fair" based on external standards, such as the contract, precedents, past performances, industry standards/baselines, and third part recommendations. We should always be open to persuasion.
- BATNA (Best Alternative to a Negotiated Agreement) – BATNA provides a reality check to determine how well alternative solutions satisfy customer needs. BATNAs must always be credible, because we may opt for them. Plan B thus should always be specific. Discussing BATNAs may also result in jointly creating new options, better than the expressed BATNAs.
- Commitment – Make your commitment with care after you have become fully informed. We should be sure that both sides know exactly what we are committing to.

Michael is a strong advocate of maintaining metrics of our project histories. Estimates can then be validated against our own performance. Software estimation tools can provide help in this area.

You can obtain a detailed article on Software Estimation and Negotiation by Michael Mah on the Boston SPIN website at http://www.cs.uml.edu/Boston-SPIN/Mah-ITMetrics-article.pdf.

Roundtable

Improving Software Estimates

Facilitated and written by Donna Johnson, President of LOGOS International, Inc.

The discussion of software estimating was well attended with a wide variety of experiences in estimating. We started with a discussion of methods for estimating. Methods discussed were:

Several people use a bottoms-up approach with various variations of that process. The bottoms up approach may be based on deliverables and/or historical data with re-estimations at milestones or results of risk management.

One participant uses COCOMO for estimating. It was suggested that people could go to the Practical Software and Systems Management web site, http://www.psmsc.com/ for information on a tool to identify the data to collect and to help collect it.

The importance of using historical data was underscored by one participant from a small company that used to grossly

underestimate projects. With historical data, the software people now do their homework and present evidence to the management. The management still may cut the estimates, but at least management does so with the knowledge that it is underbid.

Three point estimating to identify worst case, best case, and most probable case was a popular method of estimating.

Problems with estimation identified by the group include:

- Some projects are inherently different than others. Estimating efforts similar to what has been done before are much more successful. One participant expressed success in his estimates based on past experiences.
- Problems with COTS integration were a topic of great discussion. The vendor sets expectations that are not met and therefore adversely affect the delivery of products based on COTS components. Performing risk analysis to identify the incorporation of COTS products and compensating for those risks has helped improve estimates. Another problem with COTS products is their shelf life – unanticipated new versions replacing older versions, impacting software estimates.
- Estimation of fixed price projects has been a challenge. One participant ties a risk list to the cost, along with the assumptions that were made in the estimates. If the risks change, the cost changes.
- The obsession of management in wanting dates before insufficient information is known was widespread in the group, as well as having fixed end dates imposed by management. One participant said that she was able to deal with the former situation by specifying a release window based on risk around a fixed end date. The window is revisited at milestones, and it eventually converges as the project progresses.
- Over-commitment to requirements in a prototype environment was seen as a problem area. It was noted that promises above and beyond should not be made, especially in a small project, where time constraints do not allow for meeting additional requirements.

Other discussion points of interest include:

- One participant observed that 40% of risks used to be technical based; now only 20% of risks are technical-related – most of them are people, configuration risks.
- Give a sponsor a win on due dates – make sure the project delivers something that works even if it does not have full capability.
- It is helpful to have a "McGiver" on a project – someone who can spot problems way ahead of time.

Most participants use Lines of Code as a size measure. It was noted that lines of code provide a good measure for tracking faults. Lines of code, however, do not apply well in some environments, as on object-oriented projects.

Roundtable

Project Tradeoffs

Facilitated and written by Johanna Rothman

We first started talking about the kinds of project tradeoffs we make:

From a QA lead:

- People and other resources
- Can't choose project timing, projects are assigned. The strategic decision of when to do the project has already been made. Projects are queued until QA (testing) is ready. Development is finished before testing starts.

From a lead on a hardware/software combination project:

- Resources: how can people execute the work of the project? How not to dilute the resources you have.
- What has the highest priority tradeoff of all the priorities?

Johanna shared her project constraints/requirements ideas:

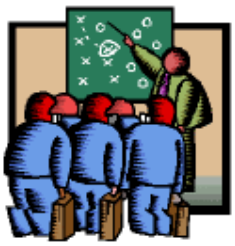
- Project constraints: cost to release, people and their capabilities, work environment.
- Project requirements: time to release, defect levels and feature set.

The constraints explain what you have available to manage the requirements. You can't accomplish more project requirements than you have available constraints.

We talked about test/development partnerships during the end of the project, the feature reduction phase.

From a software project lead:

- We sacrifice project management functions for technical progress.



From a process improvement lead:

- Trade off people. Move people from emergency to emergency

So we know there are plenty of problems. What strategies do you use to deal with the tradeoffs?

1. Use the manager to define priorities (trade off people against time to release)
2. Develop checklists instead of detailed work plans (especially for testers) (trade off time against defects)
3. Separate verification (exploratory testing) from validation (formal test cases) (trade off time against people and defects). See James Bach's web site, satisfice.com for test sheets and a project dashboard.
4. Force customer to make scope tradeoffs (trade off time, feature set, people).
    a. For agile projects, development and test are integrated

b. For traditional projects, trade off scope against test people

5. Plan the project's available time for highest effectiveness.
    a. How much description/research is enough (some testers obsess over defect description)
    b. Keep meetings brief. We discussed status meetings vs. one-on-one meetings, choosing when to have which kind of a meeting.
    c. Keep project meetings to no more than an hour. Instead of asking for what people have done, ask what obstacles remain for completing the work.

Process Improvement Roundtable

Disasters and How Process Improvement May Help

Facilitated by Judi Brodman, CEO LOGOS International, Inc.

Submitted by Dolores McCarthy, Quality Manager at Computer Sciences Corporation, Cambridge

Tonight's Process Improvement Roundtable was well attended by "SPINners" wanting to learn and chime in on the subject of process improvement. Seasoned process facilitator Judi Brodman started off by asking attendees to volunteer some of the project disasters they had experienced. There was no shortage of stories to share. Problems with requirements were heard frequently, such as poorly defined requirements, changing or late discovery of requirements, and functional requirements going unfulfilled because of a disconnect between the budget and the requirements. Difficulty with estimating cost and schedule was another area of concern, which leads to unreliable promises to the customer and loss of faith in the developer.

Judi noted that, in general, only 60% of the desired functionality actually makes it into the delivered product, because of problems such as those just stated. Part of the process should be to document requirements well and assure management and contributors understand them. It's important to know who are the major customers of the product. Requirements should always be prioritized so that the developer knows what the customer needs right away and what can wait until later, as in a staged development. Apportioning requirements is a good way to develop a product in iterations in order to give the customer what they need in an early release. Staged development with project milestones avoids a lot of rework.

For requirements discovered late in the development process, there should be a triage team to decide what should take priority at that stage of development.

Requirements may be kept in a CM tool. They don't necessarily need to be in a separate document. It's important to have a technical review of requirements, including developers and testers, to validate them. If the customer can't come to a JAD session for requirements, it's possible to send a paper prototype (screen copies) to them for review. It's essential to keep the customer in the loop. Communication must be very clear.

Another disaster example concerned a company having an inadequate business plan and no process in place. Such a company is vulnerable to unexpected changes. The company suffered the consequences when a manager the company relied upon heavily and who was extremely good at his job (translate "hero"), suddenly left and there was no plan for succession to the position. There was no documentation about how the job was carried out, and no one had been trained to take over the position. The company was at fault for this lack of planning and foresight and had to struggle trying to function without him.

Judi explained that a company needs a leader with a vision and needs to have processes and understand what the current processes are. Roger Pressman has a process assessment kit that can help a company see where it stands. The SEI CMM also has guidance available. Level 2 of the CMM is fairly easy to reach because it specifies activities to do, but not all projects within an organization have to do them in the same way. It's more difficult to meet Level 3 because the whole organization needs to come together to have a set(s) of common processes used by all.

Judi's remedies for process disasters show how it is possible for organizations to improve the way they are functioning and address the need for plans and processes to prevent such situations.

Jobs Forum Roundtable

Facilitated and written by Paul Edelman (paul@edeltech.com) and Karl Heinemann (starfarer@speakeasy.net).

Participants brainstormed a list of topics that were then discussed. These included:

- Current job openings
- The state of the job market
- Sources of information about job openings
- Methods of approaching companies
- Skills for which demand is growing
- How to prepare a resume

There was general agreement that the job market continues to be weak. Suggestions were made concerning networking and support groups. These included WIND (Wednesday is Networking Day (www.windnetworking.com), the 495 Networking Support Group (www.495NSG.com), and the Career Place at the DET Career Center in Woburn (www.careerplacejobs.com).

Recommended Internet job sites included www.boston.com, www.techies.com, www.dice.com, and www.edeltech.com.

October Meeting Synopsis

Featured Speaker– Larry Constantine

Improving Software Usability Through Better Usage-Centered Process

by Barbara Purchia, Boston SPIN chair

Why is so much software so bad, despite the time and money that is spent on "user friendly" interfaces? Where is the process going wrong? These are the questions that started Larry Constantine's talk and our journey into the power of Usage-Centered Design. The key, Larry believes, is that software should be designed almost right first and that user-centered design is not the complete answer.

There are many reasons for an up-front approach to usage-centered design. Usability testing labs are costly and focus on testing as the cornerstone of usability. Testing often comes too late, takes too long, and is relatively expensive for what is learned. In addition, the developers, not the users, make most of the decisions related to the user interface. In order to affect change, programmers need to be sensitive to the users. Once a product is shipped, addressing usability problems is very costly. In addition, people get used to the interface and balk at any changes to how they work. (The legacy legions!)

The dominant design approach today is User-Centered Design. In this approach, the focus is on the users with substantial user involvement, design by prototyping, and a somewhat variable, informal trial-and-error process. It is also a system-centric approach, i.e., the system does this or the system does that. Usage-Centered Design, on the other hand, focuses on usage or task performance and task support. Task models with selective user involvement drive it. This is a systematic process with design by modeling and correctness by design. The objective is "Simpler systems completely supporting efficient task performance and realization of user intentions; not "user friendly."

Larry walked the audience through the process he has used for Usage-Centered Design. It starts off by building simple, abstract models that can be built and hold understanding of the tasks that need to be accomplished. First, you need to understand:

- Your users – What roles do they play in relation to the system?
- Their work – What tasks are they trying to accomplish in those roles?
- Their needs – What tools and materials are need for the tasks?

Using the answers to these questions helps drive the process such that an information architecture can be developed. The process consists of the product definition, roles, tasks, contents, an implementation model, and the actual software design and then the product.

Task cases are a core differentiator in this process. A task case is a use case (one case of use) in essential form that is abstract, simplified (less means less screens), and technology and implementation independent. It is a single, discrete intention that is complete, well defined, and meaningful to a user in some role. It is not a complete job or scenario. Each task case is named with a continuing action (an "ing-word") and a fully qualified object; for example, entering a special symbol in a document. The task case is then described on an index card from two perspectives, user intentions (external requirements) and system responsibilities (internal requirements). A task case should not be more than 14 lines and should fit on 1 index card. The goal is to simplify and generalize until the task case does fit on one card. All task cases must be covered in the design. The final design is validated against the task cases.

Larry also talked about how magic happens when going from tasks to visual and interaction design. This cannot happen using iterated paper prototypes. It needs to happen at the speed of thought; something that only people who he calls "third degree wizards" can do and there aren't a lot of third degree wizards either.

This is a process that can be taught by example. He believes strongly that people need to remember how they got to where they need to be, not just the answer!

Usage-centered design has been adapted to fit development processes and practices across a wide range of products, projects, and processes. Larry has used it successfully with XP (Agile methods) and RUP (Rational Unified Process).

This was a fascinating and eye-opening presentation. If you want to learn more about Usage-Centered Design, Larry is offering a training course November 11 –15 (see http://forUse.com/seminars/ for details) and if you mention Boston SPIN, you'll get the group discount (even if you're the only one attending).

Process Improvement Roundtable

Cultural Change

Facilitated by Judi Brodman, CEO of LOGOS International, Inc.

Submitted by Marie Candela, Boston SPINner and Software Project Manager

Definition

We started with defining what culture is in an organization, determining that it includes:

- The way things get done, both formally and informally

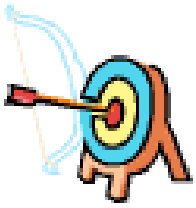
- The shared understanding of what's important in the organization, including the customer, the management, and attitudes
- The historical perspective of what is success and what is failure
- Common language

Transmission

Looking at how cultural characteristics are transmitted, we identified various mechanisms, ranging from the very deliberate and overt to very subtle:

- Company founder(s) and leadership
- Promotions/hiring criteria
- Authority and role definitions, both formal and informal
- Award and punishment behaviors
- Story telling
- Collective activities
- Corporate marketing collateral - slogans
- Company newsletters and website
- Physical layout - offices/cubes
- Office equipment
- Communication mechanisms - email/face-to-face/virtual meetings

Targets and Agents



With this context, initial targets to create a cultural change would be senior managers, funding sources, and customers. Mechanisms for leading a change are a change agent, or champion, and identified pilot teams. Pilot teams constrain the early disruption and enable the creation of a success story to support wider spread adoption of the change. The change agent must be well respected, have credibility, and manage the costs of the change.

Implementing Change

To implement change, you must understand the politics of the culture and gather information on the "problem", determining if it is truly a problem and where it is supported in the organization. You need to understand the problem, why it is like it is, and the value or benefits of the proposed change. Instituting cultural change is a selling job - you must have a sponsor, understand the reward system, and possibly have customer-influence. To be able to work with the organization you must speak their language, have their trust and respect, and break the problem down and start slowly so as to increase trust and respect.

Improving Software Estimates Roundtable

Facilitated and written by Donna Johnson, President of LOGOS International, Inc.

This roundtable was a follow-up of the discussions of the previous month's round-table on estimating. The highlights of this month's discussion follow:

- We started out with a focus on determining the role of risk in estimating and the penalty for inaccurate estimates. It was pointed out that a risk is not necessarily technical in nature, but may be a personal one – you can be fired for not completing a project within the timeframe estimated (or mandated by management).
- If given an impossible deadline by management, you can drop functionality or you must be given the tools to increase the probability of making the estimate.
- One participant described the methodology that he has successfully used for estimating:
  - Up front, the entire team votes on everything. They use flip charts and stickies and vote on how long each of the functions and attributes will take, in as fine a detail as possible. They arrive at a decision by consensus – the estimates tend to cluster together. All estimates are entered into a spreadsheet.
  - He segments the job into 3 segments. One third of the way into the job, he aims to have a working model. At that point, they look at all the estimates. They know where they are so they can better judge their estimates. They re-examine the estimates again after the second third of the project.
  - Sometimes they are 20-30% off in the beginning but that is okay because they converge as the project progresses. The estimates are calibrated to the progress made.
- Another participant's technique:
  - He (the software project manager) uses MS Project, feeding it real time information every day – each person gives him feedback
  - He tracks to the calendar time that was given him and when he detects a problem coming, he gets involved, holding people to their schedule.
  - He queries early and often, tests against calendar time.
- One participant noted that programmers are best when beating down bugs.
- A participant from a shrink-wrap company states that he drops functions if his projects run late. A program manager from a start-up company says that he expects a similar scenario. There is a list of prioritized functions to be implemented: wish list, expectations, and minimum set. A reasonable set of functionality must be established at the beginning of the project.

Someone who worked at IBM as a subcontractor related that they identify features and predict the quality (severity) of the bugs that they anticipate. They develop a matrix and measure for the level of quality that they are producing. They can predict the quality for every release. This technique provides good motivation, and it ensures that there are no regression problems. They integrate regularly and do smoke testing. They deliver all that they have to deliver. It was noted that quality is a feature that is often traded off at the end of a project if the project runs out of time.

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/

If your organization is interested in sponsoring Boston SPIN, please contact Chair Barb Purchia at bpurchia@rational.com or ask any SPIN Steering Committee member for our sponsorship 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:

- Barry Mirrer, Boston SPIN program chair, at bmirrer@alum.mit.edu.

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 2002—2003 schedule follows.

Nov. 19, '02	Capers Jones	Software Quality in 2002 – a survey of the state of the art
Dec. 17, '02	Unmesh Gundewar	Staff-less SQA
Jan. 21, '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

Happy Thanksgiving!