Letter from the Chair

As a member of Boston SPIN, we are offering you an opportunity to participate in an exciting new initiative!

We are acutely aware that our membership has been adversely affected by the downturn in the economy. We also recognize that our industry is very fluid, even in the best of times, creating a constant requirement for hiring managers to meet skilled candidates in an expedited manner.

In today's market, job seekers are applying to positions posted on the Internet and other vehicles for which they are well qualified, but often go without acknowledgement. Many have successfully used the networking time allotted in our programs to introduce themselves to other members who would benefit from their services. Others have been less successful in their networking efforts, while still others are uncomfortable with networking at all.

Hiring managers are being inundated with resumes and are steadily being approached in networking situations by people who are looking for work, but do not meet their needs.

As a service to our community, membership, sponsors and industry, we seek to expedite the process by which hiring managers are introduced to qualified applicants from within our membership as openings become available. We also seek to help our members improve their job-searching skills. Towards these ends, we are taking the following actions:

1. A roundtable will be established at the March meeting specifically to discuss employment issues - resumes, available resources, networking tips and the like.

2. In mid March, we would like to meet with prospective hiring managers from amongst our membership, outside of the SPIN calendar, to discuss tangible ways through which we can meet the objective stated above. Anyone interested is encouraged to attend regardless of immediate needs. The goal of this meeting is to determine a plan of action that will kick off at the general March meeting.

Should these initiatives prove popular, they will become standing elements of the Boston SPIN venue.

Please respond to this announcement if:

You would like to offer suggestions as to the focus of the roundtable.

You are a prospective hiring manager willing to participate in the planning discussion. Indicate your company name, title, and day/time of week most convenient. We will return your response with additional information.

You are looking for a new position. Indicate the desired position within your response. Please do not include a resume or profile.

We hope that you are as excited about these initiatives as we are! Boston SPIN is much more than listening to a speaker. We strive to be an alive, vibrant organization genuinely committed to your professional growth and prosperity. Thank you in advance for your suggestions and participation.

Regards,

Linda McInnis, Chairman, Boston SPIN
Richard Green
Michael Brother
This article is about Extreme Programming and Triage for 2 reasons. First, if I had to pick the top benefit that XP has yielded my team, it's the natural "requirements triage" it forces on both management and developers. And secondly, a widespread, growing problem in software development is the disconnect between the software developers and those who sponsor the development activity. Mistrust leads the sponsors to inflate the importance of features requested and to demand completion in a very short time. They fear that it’s the only way they will end up with something usable in a timeframe that they can live with. The solution to this problem is for both sides to acquire some negotiation skills.

The XP mechanism for triage is principally the "Planning Game" practice, and you can use this to break the "chicken or egg" cycle of SW development problems. We all have seen software projects sabotaged at the outset by unrealistic goals. Management has the power to demand x amount of functionality in y weeks of effort, and anyone who says it can't be done is viewed as a loser. So the project starts with the SW developers saying they'll do all they can to make the goals. Shortcuts are taken, causing bugs, delaying the work, causing more mistrust between management and the development team. And the cycle goes on.

Extreme Programming gets at the root of this insidious problem by giving the development people an acceptable way to tell management when something *really* cannot be done. It also gives management a way to keep the developers from running away with the project. We've all seen projects where the feature set gets driven inappropriately by the developers - they'll decide to add cool features because they expect it to be easy to do, or they are sure the users will want these extras. Mistrust leads the sponsors to inflate the importance of features requested and to demand completion in a very short time. They feel that it’s the only way they will end up with something usable in a timeframe that they can live with. The solution to this problem is for both sides to acquire some negotiation skills.

I'll give an example of a Planning Game session that occurred with my own team. There were 6 features that 'Business' wanted - we had informally discussed them beforehand. Prior to meeting with our project manager who was acting in the 'Business' role, we looked at the code to map out a detailed design of how we'd implement these features (taking a day or 2 at most). We didn't write up any extensive design info. We just noted which existing functions and modules would be changed, and what new ones would be created. We also noted any changes to data structures, especially those shared between modules or other interfaces. Then we assigned points estimates to each feature. Our team had been doing 4.5 points per week in previous work, and so, depending on the due date, we wanted to commit to the same level of effort.

When we met with our project manager, he wanted the release by a certain date - there would be only 7 work days to do these features. The points total was 7.5 - clearly impossible by our latest demonstrated capability. So we asked him to pick a feature that could be delayed to the next release. We could have proposed a change to one of the features to make it less effort, but in this case it made most sense to delay one feature. That reduced the points total to 5.5, and 7 days is 1.4 work weeks, so we were signing up to 5.5/1.4 or 3.9 ptSWk. The resulting agreement we made is listed below.

Statement of work, in order of priority:

<table>
<thead>
<tr>
<th>Who</th>
<th>Points</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL/NS</td>
<td>2.5</td>
<td>Automatic Exposure Control (AEC)</td>
</tr>
<tr>
<td>NV</td>
<td>0</td>
<td>Change Filtering Pixel from 720 to 330</td>
</tr>
<tr>
<td>SL</td>
<td>0.5</td>
<td>Change Filtering to Zero-out spectra if insufficient samples obtained</td>
</tr>
<tr>
<td>NV</td>
<td>0.5</td>
<td>Re-enable Filtering for Field Use</td>
</tr>
<tr>
<td>TL</td>
<td>2.0</td>
<td>Add Reliability Data Check for PLD info</td>
</tr>
</tbody>
</table>

**TOTAL** 5.5

The points estimate is not function points. It's a measurement described in the XP literature. It's not so important what estimating technique you use as long as it is something the developers know how to apply and can use confidently (and quickly) to bid the work. One feature is estimated with zero points because it was just a change to a #define in the code and could be tested simultaneously with one of the other features.

Another thing to note is that 'Business' cannot tell 'Development' to take design or testing shortcuts. The developers *accept* their responsibility; it is not *assigned* to them. Given what is known about the planned final feature set, the developers can put the appropriate design foundations into place. XP places a lot of pressure on you to put them in place incrementally however. If you need a database to support one feature, you can bid a lot of points to fully do the database and then put the feature on top. But if
there is a way to just put in as much database as you'll need for a stripped-down initial version of the feature, then you can deliver more business value for less effort. It may turn out later that the feature isn't as needed as 'Business' thought it would be so the stripped-down version of it is fine as is. Or maybe they change their mind about it altogether and drop it. The code is left in good shape if you used a legitimate (from the SW design standpoint) way to implement an abbreviated version of that feature.

The beauty of this sort of triage is that it harnesses the creativity of the developers to do what they know best: how to take the present code base and modify it to add or change features in a way that does not introduce kludges, or other disjoint interfaces that can generate bugs. It likewise frees 'Business' to concentrate on what they know best: how to position the product feature set to appeal to the targeted users. Even if you think your management doesn't know how to do their side of this bargain, you'll be ahead if you set up a clean division of labor where your software team is delivering on its promises. If it later becomes clear that your marketing folks had you do the wrong mix of features, let it be dealt with at a higher management level.

The Planning Game is very effective as a negotiating tool. It is very hard for developers to push back against unreasonable schedule demands because they usually have nothing management can "see" - they are just stating their opinion that the time's too short. They are right but being right is not enough. They have to show the other side some convincing evidence. Usually there is none because there aren't any metrics to fit this team here and now. Our team started using the Planning Game by simply doing the points estimates on our own for a release we had signed up to do already. When that release was complete we then had a comparison of our estimated points and the actual points completed in x working days. That number was used as our points per week "speed" for the first official try at using the XP Planning Game process. That was way easier than a big production to generate SW metrics and history for the whole company. Something like that is hard to sell to management and can get bogged down too easily. The whole spirit of XP is to use your knowledge to do something as simply as possible that will give value.

For negotiating, when you can say to your 'Business' player that you think their due date is too short because the folks who will implement the features have already looked at how to do it and they estimated it as being the same effort as the previous release which took 3 weeks, then it is very hard for them to counter that. How can they insist it will only take one week to do this new work? Even if the business player is the former software lead person and says they can do it in a week you can simply offer to walk through each of the individual feature estimates and the notes showing the extent of the changes. If they still claim it can be done in one week, offer to let them do it. At the very least you have good ammunition to insist that they agree to cut scope a week into the work, and you will implement the most important feature first.

The hardest thing in using the Planning Game is to make sure you deliver every release on time. You have to cut scope to do it. Cutting quality is out. Nobody wants buggy code, no matter how urgent the situation is so forget that one. In some Planning Game sessions I had a hard time getting Business to say that any feature could be sacrificed, so I would draw a line across the feature list and say to them "Suppose the due date arrives and we have all the features implemented except for the ones below this line? Will you refuse to take this release?" The next question is "OK, so you've said every single feature is critical - is there even one that could go below the line?" If they say no (or won't move enough of them to non-critical), then I change the order of the features in the list and say I think I've ranked them in the order of importance. Inevitably, they'll rearrange them. That's fine. I get a commitment as to which is the most important, the next most important, etc. I then say that our metrics tell us we can safely get the first 2 features done (or whatever the actual number is), and getting the last 3 of them in is going to be high risk. We'll do all in our power to work diligently and test thoroughly and if everything goes without a hitch we may be able to get the other 3 things in but, if not, then I want to meet again with you on x date to re-prioritize. This has gotten people to go along. If there truly is no chance of getting those extra 3 features in I don't hold out hope of it; I'll say we will do our best but since our metrics say its such a long shot, then I insist that we look again at the situation on day x. How can they refuse to? Now the onus is on them to say we're not working hard or we're doing unnecessary stuff. We can show exactly what we're doing and why because of the detailed design we did as prep for the Planning Game meeting. On the good side, it gives management an incentive to make sure our time isn't wasted by administrivia or needless interruptions.

The biggest reason our software got completed and is in field use now is that we practiced requirements triage. We also did sufficient design before starting to code. And we used iterative SW development. I believe those 3 are all strong contributors to success, but without the triage the other 2 wouldn't have been enough. Both Business and Development have to practice triage. I have a list of all the stories that were ever seriously considered for this SW, and it is at least as long as the list of stories we implemented. Some of the leftovers are features I was sure we needed. For instance we have a run-time trouble log and it's in RAM. As long as we don't have a bug that messes up the communications link we can retrieve the log. But we should store it in Flash memory so we can retrieve it no matter what. Well, that effort was always too large to get prioritized. Business understands the risk we run and they have made a judgment to live with it. This is proper - they get to decide what functionality they need and are willing to pay for.
When the release work is underway, it's important to implement the most important feature first. I mean the one 'Business' has said is most important. If unforeseen problems occur then you want to be in a position to cut scope and still deliver business value. Fully test each feature as it is completed. If you have completed 5 of 6 requested features and time's up, you can offer the release to 'Business' or they can decide the date is movable, and re-plan. If they accept the 5-feature release after having insisted that no feature could be omitted, then they lose a little credibility. They will pause before bluffing again. It takes a few iterations to build trust but it's well worth doing.

Now that I have given you the Planning Game rules, I want to note that it is just one of the twelve practices within XP. They all work to support each other. For example the existence of a coding standard enables collective ownership of the code, which in turn supports pair programming. A coding standard, collective ownership, and pair programming are 3 other XP practices. In my opinion a team would have difficulty using the Planning Game practice if they didn't already have collective ownership of the code established within their group. Also I didn't go into ranking stories by risk here. That is also part of it. I'll refer you to the literature for this.

Recommended website: There is info on coaching XP teams. (I'm not connected with this company - they just have good XP literature):
http://www.objectmentor.com/default.shtml

Recommended website: Ron Jeffries' site at http://www.xprogramming.com/

Committee Spotlight

Email Happens

Rick Brenner is principal of Chaco Canyon Consulting, providing consulting, training and coaching to problem-solving organizations. He publishes Point Lookout, a newsletter of tips and insights about working in project environments. Visit his Web site at www.ChacoCanyon.com.

Email is great for making appointments or distributing agendas. It's less effective for resolving conflict or discussing solutions to complex problems. Yet we're still surprised when an email discussion goes awry. What are the limitations of email? Where do they come from? And how can we use email safely?

Jenn was puzzled. Benson, the only one left who could keep the creaky old payroll system alive, had just requested a transfer. At first, Benson had wanted to work with Fran on the new payroll system. Travel budgets being as tight as they were, Jenn had accepted Fran's suggestion that she and Benson work out the specs of the new system in email. The next thing Jenn knew, Benson wanted out of the group. What went wrong? Email.

There ought to be a bumper sticker: Email Happens.

Email is great for making appointments, distributing agendas, and even brainstorming. It's less effective for discussing complex problems, soothing hurt feelings, and resolving conflict. Yet we're still surprised when email discussions go awry. What are the limitations of email? And how can we use email safely?

When compared to face-to-face communication, Email has important limitations. Recognizing them is a key to using email safely.

Save tender subjects for the phone or face-to-face

As you type, you can't tell what reaction you're getting. When you talk with people face-to-face, you can detect reactions — facial expressions, body language, breathing, voice tone, even silence. You can spot trouble, and you can make mid-course corrections. In email, you can't.

Be brief The cycle time of email exchanges is long. Even when email is fast, the cycle might be ten minutes or longer. In face-to-face communications, the cycle time can be less than a second. To compensate for the delays of email, we send longer messages, which creates risk. If you offend someone face-to-face, you find out quickly, and that limits the damage. In email, we sometimes exchange whole screenfuls, and if someone is offended early in a message, the offense only grows with each paragraph.

Use smileys and a conversational tone When most of us learned to write, we learned formal writing — proper grammar, fancy vocabulary. In email, that style sounds stiff, and creates barriers between you and your readers. To sound more conversational, use short sentences, contractions and sentence fragments. Like this. Beware of sarcasm—you're readers might not get it. Use smileys to make sure.

Interpret email in the most positive way possible The sender is probably saying something nice, rather than delivering a cleverly disguised insult, because—sadly—most of us no longer write well enough to disguise insults cleverly. And I don't mean that in an insulting way.

Email is to true communication what fast food is to fine restaurant fare. I'd go for fast food at the end of a bike ride. On Valentine's Day, or for an anniversary, fast food just will not do. Don't even try it.

Copyright © 2002 Richard Brenner.
The Need for Process on e-Business Projects

This article appeared in Cutter Consortium’s March 2001 e-Project Management Advisory Service as an Executive Summary under the title “Applying the CMM to e-Projects” by Donna L. Johnson. Copyright 2001 by Cutter Consortium, 37 Broadway, Suite 1, Arlington, MA 02474-5552, USA, Web address: http://www.cutter.com. It is reprinted here with Donna’s permission.

Project Management is arguably the single, greatest challenge facing e-business projects (e-projects) today. Differences between development on e-projects and traditional projects are quite pronounced -- the life cycle of most e-projects is shorter and more compressed; the technical challenges rival those of the large systems of the 1980’s; and an increased array of stakeholders (such as sales, marketing, production, engineering, warehousing, customer support, and outside parties such as subcontractors, ASPs, WEB hosting providers, and public carriers) create conflicting demands on an e-project. An ad in the Winter 2001 IBM Software Catalog succinctly captured the challenge:

“At Four-thirty on a Sunday morning, a Web Site fries. At eight-thirty a.m., in a conference room, the IT Director wants to know what happened. The server vendor says it’s an app problem. The front-end vendor says it’s the database. The database vendor insists it’s the servers. All told, there are twelve vendors. Twelve. The IT director wants to know exactly who’s responsible for making this stuff work together. After a long, ugly pause, the guy next to her whispers into her ear, ‘You are.’”

Unfortunately, the beleaguered IT Director has it easy compared to an e-project manager -- she just has to keep the system working. The project manager, on the other hand, must also plan, review, build, revise, manage and deliver the working product. He or she deals with the same 12 component vendors, as well as innumerable tool, service, and certification vendors, an ever changing development staff, shifting marketing requirements, budgetary issues, and, last but not least, the beleaguered IT director.

Within this e-project environment, there is also an ever-expanding domain of new technology where whole new fields of endeavor are born virtually overnight. The accelerating expansion in technology, fed by exponential Internet growth, requires practitioners in software development to acquire multiple skill sets. Interestingly, although facility and expertise with a daunting array of skills (such as XML, WML, J2EE, JavaBeans, embedded Java, RUP, telematics, VoIP, and convergent applications) is expected of job applicants, software engineering skills are seldom mentioned as a prerequisite -- quite remarkable given that the discipline of a structured software or systems engineering background is needed more than ever in the e-project environment. This need is readily apparent when one examines the characteristics that distinguish an e-project from a traditional project:

- The labor pool of software people is largely inexperienced in the distributed Web domain, and the repercussions of such inexperience often create havoc with e-business applications. Developers making changes to code simply do not understand that they should look for system implications; when agreeing to requirements, managers and developers alike often do not grasp the systems impact; and, when implementing requirements, designers often fail to first analyze key feasibility and performance issues. Without training or processes established to compensate for the lack of experience, the quality of Web applications will remain unacceptable.

- The need to beat the competition to market or to meet customer’s expectations for delivery in Internet time has compressed software schedules to the extent that there is virtually no room to make mistakes – the software has to be built correctly the first time, with high quality, before funding runs out. Managing risks, reducing rework, and effective planning of the project are all necessary steps to bringing the project to its earliest possible completion.

- The volatility of requirements, especially in Internet applications, is an accepted way of life on e-projects. The technology is changing at a rapid rate, requirements are deliberately left vague to allow for response to technology and business model changes, and the large number of e-project stakeholders creates the opportunity for many changes to occur. Without a process for introducing change into e-project development efforts, excessive thrashing of requirements and development efforts can significantly impede product delivery.

- The interconnectivity among the myriad applications that support an e-business application creates a coordination nightmare. Interfaces have to be understood, adaptors have to be built, requirements have to be agreed upon, problems have to be resolved within time constraints imposed by service-level agreements, and system integration and deployment have to be conducted without impacting business operations. Exacerbating this situation are the interoperability issues. Seldom are systems compatible, and rarely are components truly plug ‘n play. To address this assemblage of disparate demands, the project manager needs efficient mechanisms to maintain communications channels between all the parties involved in an e-project.

- Web applications have worldwide exposure – hence, risk -- via the Internet: seldom are today’s e-projects deployed in a small user community or localized enterprise. Typically, they are targeted for widespread access via the Internet, especially since privacy concerns can now be addressed through the use of virtual private
networks, IPSec, and other solutions geared toward the Internet. But such global deployment of e-commerce applications can expose a company to an assortment of woes, ranging from lost revenue and disenfranchised users to lawsuits. For this reason, quality is of paramount importance -- the very survival of many e-commerce companies depends on the smooth, reliable operation of their e-business. Improving product quality while minimizing time to market and preserving functionality creates challenges for even the most seasoned Project Manager.

Taken together, these e-project characteristics argue strongly for an aggressive approach to adopting structured development and management processes. Despite the widespread skepticism concerning the applicability of formal process models to today’s ever-changing e-project environment, process is an absolute necessity for addressing the problems now being experienced in that environment -- rather than posing a burden to e-project development, process offers a solution to their problems.

The Software Engineering Institute’s Capability Maturity Model® for Software (SW-CMM, or CMM®) is a time-tested tool that can provide the structure necessary to control e-project development environments. The CMM provides a good framework for introducing solid, proven software development practices into an organization and onto projects. Its value has always been the fact that it lays out a progression of steps to take toward improvement. Its added benefit is that it is prescriptive rather than prescriptive -- the model can be adapted to a wide variety of development environments, including the e-project environment. Its adaptability has already been proven for the small project environment. Its particular appeal in the e-project environment is its heavy emphasis on project management - the biggest challenge on e-projects. It is a proven model, there is return-on-investment data available on its efficacy, it is widely understood, and there are tools that support its application. All these facts leave the CMM poised to help the Project Manager meet the challenges of e-project software development.

® CMM and Capability Maturity Model are registered trademarks of Carnegie Mellon University in the U.S. Patent and Trademark Office.

---

**February Meeting Synopsis**

**How Much Process is Enough?**

Speaker: Linda McInnis, BostonSPIN Chair
Noble Associates, Inc., Albany, VT 05820

Linda’s presentation was based on her writeup below.

As software professionals we are constantly bombarded with new processes and methods for shipping good enough software and sometime we get caught up in the whirlwind of What's new, What's in fashion – these processes often make the well worn claims of faster, better and cheaper. But are they faster or cheaper or better?

In this article I am trying to show some rules of thumb to help you determine whether your process is on track or need of some modifications.

First, what is process?

proc·ess
1. A series of actions, changes, or functions bringing about a result: the process of digestion; the process of obtaining a driver's license.
2. A series of operations performed in the making or treatment of a product: a manufacturing process; leather dyed during the tanning process.
3. Progress; passage: the process of time; events now in process.

Software Development process is the series of actions it takes a development team to produce good enough software to be marketed and accepted by it's client base. Current development processes range far and wide. Does it matter which one we use? Realistically, it doesn't matter if it is working with your team to produce good enough software or hardware shipment on the designated schedule. All successful software development processes share the following elements:

- Accountability
- Shared Responsibility
- Planning
- Active Participation
- Measurement
- Teamwork
- Active Management

You may have to use certain methodologies to pass legally mandated performance criterion such as FDA regulations, Product liability statutes, or other industry-mandated methods.

*How do we know it's working?*
I'm going to give you a series of Rules of Thumb to help you decide whether your team is on track or not.

**Rule of Thumb #1** - “A Process is working if it’s easy to do the right thing,” Marco Ocana, manager, author and software developer

**Rule of Thumb #2** - A process that is not in alignment with company vision or mission is doomed to fail.

**Rule of Thumb #3** - You will get behavior that is rewarded. Beware of what you reward.

One of the primary criteria is really how easy is it to get productive work done on your project. In each of the following areas of your project ask the questions about your team and see where some methods might need to be evaluated.

**Management**

Is management aware of your process?
Do they support it?
What are their expectations for process?
Is there a schedule for the project?
Does the team regularly meet it?
Are there written requirements?
Is there a written project plan?
Is the project fully staffed?

**Process**

Is your process documented?
Is it accessible to all members of the team and is it visible?
Do Process team members understand the objective of the software development effort?
Is your process evaluated regularly?
Is it updated regularly?

**Developers**

Do they know what tools to use?
Do they know where the code repository is?
Do they know how to use the repository?
Is there a development schedule in place that includes test time?
Do developers respect the other members of the team?

**Risk Management**

Does a plan for disaster recovery exist and is is used by the team?
Is the legal department aware of the project?
Have they assessed any potential reliability or liability concerns?
Have tool suites been assessed for contractual use problems?

**Configuration/Release Management**

Can the product be compiled and built upon request?
Has the system been tested within the first month of the project?
Do certain developers always have to "help" build?

**Quality Engineering**

Do they have the tools sets they need?
Do they have adequate personnel for the current development?
Do they have a plan for a schedule crunch?
Is there a metrics plan that is a criteria for shipping?

**Customer Relations (Support)**

Do they know how to deal with requests from beta sites?
Can they work as part of the team to help with prioritization of requested changes?

**Technical Documentation**

Do they have documentation milestones that correspond with development and testing?
Do they have a plan for Release Notes?
Do they work closely with Support and QA on a regular basis?
Do they understand they need to report bugs they find?

**Marketing**

Do they understand shipping criteria?
Did they generate or review the requirements doc?
Do they understand the impact of changes on the process?
Have they explained what the market window is?
Have they explained the market opportunity to the whole team?

**All**

Does the whole team meet regularly to report status?
Does the team display professional, pleasant interaction?
Is there sufficient feedback in the process to allow an average person to know how they are performing?
Do most team members know whether the team is on track or not?
Is the process repeatable?
Does everyone know how the decision to ship comes about?
Do managers regularly circumvent the process to ship their software?
Do you know what the cost to the company is in re-work?
Is the process forgotten as soon as the product ships?
Do you have a post-ship project evaluation?
Do you have the same team for the next release as the previous release?
Does everyone understand the cost of missing the market opportunity window?
Are the following criteria in performance reviews: effective communication, team work, adherence to schedule and quality?
Do the process rely on 75% of a 40 hour week and allow for sick days, vacations and emergencies?

Are you getting "No" more often than "Yes?"
We have some problems. What do we do about them?

The kinds of problems that most teams run into break down into the following areas. Here's some suggestions on how to address them.

**Scheduling**

If you've got a scheduling problem, gather the team and try to find out why. Then revise the schedule in small chunks with more frequent milestones and review with the team at least twice weekly until project completion. This rebuilds integrity in the scheduling process and the team regains confidence in itself.

Schedule forward based on the experience of the people that will do the work. Know what the needs and wants are in the product specs and be firm on the schedule.

67% of all projects fail to deliver either features, on time or at all – plan to be in the 32% of successful projects.

**Personnel**

Not enough? One too many? Poor quality? These are all problems that are part of a Manager's job and you have to manage them. You have to say "No" that you can't do a project when you don't have the people to do it. Try as you will, if you don't have people or you have the wrong people, the work isn't going to get done.

**Requirements**

The lack of requirements is epidemic in software development today. Give your team a break and make sure that you have a set to begin the project otherwise you'll spend too much time arguing between development and QA or QA and Marketing or Tech. Doc. Saves time and money.

**Technology**

There are no silver bullets. Technology quite often does not perform as advertised. If you're having problem with a new tool – spin out 1 person to figure it out – not the whole team. Modify the schedule accordingly.

Technology can, however, help you communicate with distant members of your team.

**Communication**

The root of all great teams is communication and respect. If people are not talking to each other, don't accept this. Give people a chance to show their greatness and it will happen. It's hard to objectify people you have to see every day and most folks will accommodate eccentricity when they see good sides of people they don't like.

Make opportunities for people to be great and to show people's good sides. Catch people doing things right and make sure everyone knows it.

**Management Support**

If you have educated your management as high as you can go about the cost associated with bad implementation and process and you keep repeating it, and nothing changes, you will have to make a personal decision about whether this is the right company for you to be in.

**Bottom Line…How much process is enough?**

Whatever it takes to ship good enough software on time.

**February Book Club**

“Getting to YES: Negotiating Agreement Without Giving In”

by Roger Fisher, William Ury, Bruce Patton (for the Second Edition)

Facilitator: John Brtis, The MITRE Corporation

The group walked through and discussed the major points of Getting to Yes. Five major recommendations for successful negotiation were touched on.

1. Focus on Interests, NOT Positions

Bargain over interests of both sides, not over positions. Interests define the problem not the solution. There may be many possible solutions to the problem. One way of moving beyond the solution is to ask, “Why are we proposing this solution?” Don’t stop with only one interest. Each side will likely have multiple interests (the most powerful interests are basic human needs).

The group discussed how this separation is conceptually similar to the effort that needs to be made to keep customers focused on requirements rather than how the problem will be solved or how the code will be built.

The group had found it useful to use storyboarding to objectively identify how things are done now and evaluate the points of pain that are experienced. This is done before solution finding is entered in to.

2. Invent Options for Mutual Gain

Separate the inventing of options from the deciding of which option is the best. This will allow more possibilities to be identified. Always look for mutual gain, a solution that meets the interests of both sides. A compromise solution is not a good solution.

The group discussed successes they had had with brainstorming techniques in achieving these goals. Tactics that seem to work are suspension of criticism, yellow Post-Its on the wall, then grouping items to identify categories, organization, and order.
3. Insist on Objective Criteria
Deciding on the basis of will is ineffective. Develop and agree on objective criteria. To do this, frame each issue as a joint search for objective criteria. Reason and be open to reason, but never yield to pressure.

4. Separate the People from the Problem
Negotiators are people too. Keep in mind that every negotiator has two kinds of interests: in the substance of the negotiation and in the relationship with the other negotiator. Separate the relationship from the substance; deal directly with the people problems.

5. Develop Your Best Alternative to a Negotiated Agreement (BATNA)
Your negotiating will be ineffective unless, going into the negotiation, you know your best alternative to a negotiated agreement (BATNA). This will allow you to intelligently determine the level of compromise you can consider while still benefiting from staying in the negotiations. It can also be very helpful to estimate your opponent’s BATNA.

February Roundtable
Process Measurement Roundtable
Facilitator: Michael Brother

After a mad scramble to find enough chairs, we started the session by discussing what we wanted to learn about process measurement. The topics covered a broad range of measurement issues:

- Reducing a lot of data to manageable metrics
- Using data to move towards process
- Minimizing collection effort and maximizing communication
- Defining units of measurement, how to be clear in what you are measuring

Everyone wanted to use data to drive effective actions. Many had effective data collection and were struggling with how to best use the data. We built a framework to show the transformation of data to information to knowledge and then to action realizing that data by itself cannot produce results. Linking the data to meaningful goals provides an effective link to action.

We then reversed the process to decide how to define the least number of metrics to get results. By starting with a goal, we can develop questions that help us understand if we are achieving the goal. Next, we can define the metrics that answer the questions. Goal-Question-Metric can be used for strategic goals, like increasing productivity, or for more tactical goals, like proving that low bug rates increases profits.

Using bug rates and profitability as an example, we brainstormed different ways to turn the bug tracking data into knowledge. Well defined units of measure became critical at this point. We agreed that different audiences needed different views of the same data to understand the issues. Senior management needed to know the dollar impact, while development managers were more interested in saving time and effort. By measuring effort to find and fix a bug, we can present a compelling story to the development manager. If we know who is doing the work, we can easily convert effort into dollars so a single set of metrics provides value to different audiences.

We talked about how the cost of bugs varies, and once again needed to clearly define our terms. Cost is very different for customers and developers. Our discussions focused mainly on internal costs to find and fix. We agreed that customer impact should drive the prioritization of bug fixes. Often a low engineering impact problem has an intolerable impact on the customer and vice versa.

The discussion was lively, the participants interesting, and the topic was broad.

March Meeting
Roundtable Programs
6:00 - 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: Retaining Staff in an Uncertain Economy
Facilitator: Johanna Rothman

When the economy is good, it's easy to retain staff with monetary incentives and rewards. However, when money is tight, and salaries are holding even, what do you do to retain staff? Do you change your management style? Do you use non-monetary rewards, such as appreciation dinners, training, conferences? Do you use something else?

Bring your questions, what you've tried, and we'll discuss how to retain technical staff in an uncertain economy.
Roundtable 2: Job Seekers  
Facilitator: Michael Brother

The rules for finding a job have changed. With so many skilled people out of work, finding a job is harder than most people can recall. A good recruiter is no longer the only answer. Using networking, the recruiter's most powerful tool, is now a required skill. This roundtable is an opportunity to share your experiences and insights on job hunting today and networking. We will also explore how Boston SPIN can help in your job search.

For those of you actively networking already, this is a how-to session and not a networking session. Remember, networking is like a smile. It is better when shared.

Bring your resume, experiences, and questions. You will leave the roundtable with a better understanding of networking and how to use networking to energize your job search.

Roundtable 3: Process Improvement - How to persuade your sponsor, management chain, peers and subordinates  
Facilitator: Judi Brodman

This roundtable will address the problems and solutions for each of the subtopics. Bring your experience in the problems you've encountered and solutions that worked (or didn't work) when you were faced with persuading your project sponsor, management, your peers and your subordinates in the benefits of good process.

Roundtable 4: Software Testing SIG  
Leader/Host: Paul Piper

This special interest group (SIG) provides an ongoing forum for speakers, presentations, demos and panel discussions of software testing topics prioritized by Boston SPIN members and guest participants. Contribute your experience, questions, successes and challenges and join the forum discussion.

Monthly Boston SPIN Book Club

“Are You Lights On (How to figure Out What the Problem REALLY Is),”

by Donald Gause and Gerald Weinberg

Facilitator: Barbara Purchia

This powerful little book will help make you a more effective problem solver. Are You Lights On helps you figure out what is really wrong, and also what is just as important, what solutions are viable options. It provides humorous anecdotes on problems you may have already encountered and then provides helpful and valuable advice to help with these problems. Anyone involved in product and systems development will appreciate this practical illustrated guide, which was first published in 1982 and has since become a cult classic!

With their characteristic clarity and insight, authors Don Gause and Jerry Weinberg instruct on ways to improve your thinking power. Are You Lights On offers such insights as “A problem is a difference between things as desired and things as perceived” and “In spite of appearances, people seldom know what they want until you give them what they ask for.”

This book describes four steps to figure out what the problem REALLY is in a:

1. Identify the true problem
2. Determine the problem's owner
3. Identify the source of the problem
4. Decide whether or not to solve it

Come join us on March 19 and let’s talk about if and how Are You Lights On changed the way you view the world.

March Main Program

Software Process Improvement and e-Business: an Oxymoron

Speaker: Donna Johnson, BostonSPIN SEI Liaison

When discussing software development in an e-business environment, industry literature emphasizes the need for e-business projects (e-projects) to be flexible and responsive to change. In order to meet this need, some people advocate that process has no place in an e-business environment; while others recognize the need for process, yet advocate the use of light process methodologies, such as Extreme Programming (XP), in lieu of heavy methodologies, such as the Capability Maturity Model for Software (SW-CMM or CMM). This presentation examines the characteristics and environment of an e-project to determine the role that software process, including the CMM, can play in solving many of the problems being experienced by e-projects. The speaker interviewed 18 organizations engaged in the development of e-business software, many of which are using structured processes in their development efforts. Examples will be drawn form these organizations to show the benefits that they derive from following structured processes.

Donna Johnson is President of LOGOS International, Inc. Since 1991, she has been consulting, conducting training and assessments, and performing research related to software process improvement (SPI), specializing in process improvement in small organizations and on small projects. She is co-author of the book, The LOGOS Tailored CMM for Small Businesses, Small Organizations, and Small Projects, which is used in organizations worldwide. Her other areas of focus include the return-on-investment for CMM-based SPI, project management, requirements analysis, and e-business software development. She has published articles in leading industry magazines and for Cutter Consortium and regularly presents at
conferences on SPI issues. She is a co-founder and past Chairperson of the Boston SPIN.

**Upcoming Meetings**

April 25, 2002 (Thursday) – NOTE NEW DATE
Joint ASQ/SPIN dinner meeting, Tom Demarco, guest speaker will discuss his new book, “Slack, Getting Past Burnout, Busywork and the Myth of Total Efficiency.” Watch our web site for details.

May 21, 2002
Speaker, Tim Lister
Book Club, Peopleware (Productive Projects and Teams), by Tom DeMarco and Tim Lister

June 18, 2002
Speakers, Judi Brodman & Steve Hannigan, Achieving CMM Level 2 and Beyond
Book Club, Surviving the Top Ten Challenges of Software Testing (A People-Oriented Approach) by William Perry & Randall Rice

**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.

**Book Purchase Program**
Boston SPIN is bringing you more convenience for your book club selections. You can now order Book Club selections and your purchase benefits SPIN. We receive a percentage of every purchase made by linking from our site.

Book Titles are linked to their reviews on Amazon. Go to the This Month's Meeting or Book Club Schedule pages on our website where you can read an abstract of this month’s Book Club. From there it is just a single click to the Amazon.com website to purchase it!

**Future Programs**
We welcome your suggestions for future Boston SPIN programs. Program suggestion forms can be found on our web site [http://www.bostonspin.org](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](mailto:Boston_SPIN@yahoo.com).

**SPIN 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](http://www.bostonspin.org)

**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](mailto: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](mailto:brodman@logos-intl.com) Sheila Lynch, Co-editor of In-the-SPIN [salyuchi@mitre.org](mailto:salyuchi@mitre.org)

To receive notification of new In-the-SPIN issues and Boston SPIN specific notices, send email to: [withall@mediaone.net](mailto: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](http://www.bostonspin.org)