

In-the-SPIN

Newsletter of the Boston@SPIN

Issue 54 Winter 2005

Editor: Judi Brodman

From the Editor

By: Judi Brodman, Editor



First, congratulations to the both the Boston Red Sox and the New England Patriots!!!! Thanks for the memories....



Now, on to SPIN business - this issue contains a very interesting and informative article by one of our SPIN members on using the SEI's Team Software Process (TSPSM). He discusses the use of the statistical technique called Capture-Recapture – a method used to estimate the remaining software defects in a unit of code. His analogy with fishing is terrific! Happy fishing!

Also included in this issue are the results of the November and December Roundtables (RT). The RTs continue to be very popular and the sharing of experiences and expertise continues. If you haven't attended one, you really should.

I'm always searching for new articles to publish in our Newsletter. Write up your experiences with methodologies (Agile, Scrum, etc.), Software Process Improvement (working towards achieving a Level, Appraisal experiences, etc.), estimation and metrics (always popular discussions). Send them to me and share your experiences with our members. If you have questions that need answering, send them to our SPIN Doctor and have experts answer them for you.

If you have any comments or questions about our *In-the-SPIN* newsletter, please let me know - <mailto:brodman@logos-intl.com>.

Happy, healthy, and prosperous 2005!

Judi Brodman

From the Chair

By: Barbara Purchia, Chairperson

Our Steering Committee has had some changes. Caroline Starita has stepped down as our Roundtable Chair. She initiated the Roundtables when we were meeting at General Dynamics, over 7 years ago, and did a fantastic job of managing them over the years. We thank you Caroline!! Dolores McCarthy has volunteered to take over that role and is busy setting up this year's roundtables with interesting topics and facilitators. Thanks Dolores for stepping in.

John Britis had to step down as Program Chair due to work commitments and Donna Johnson, our first SPIN chair and currently our SEI Contact, volunteered to fill this role. John put together a great program last year and Donna is keeping up the tradition this year. Thanks to both!

I'd also like to thank the following volunteers and Steering Committee members who help make each meeting special: Anna Glebova and Vlad Slezin, our Greeters and Feedback collectors. Anna is also our Webmaster and she does a wonderful job keeping our site up-to-date. Ron Kay, a Member at Large, is our refreshments person and brings the goodies, sets up and cleans up after the meetings. Sheila Lynch, our MITRE Liaison, makes sure that everything is set up and ready for our meetings, roundtables, and refreshments. Jon Leer, our Program Publicity Chair, sends out our meeting announcements to different venues like Mass High Tech and 128 News. Rick Brenner, our Vice Chair and Sponsorship Chair, has secured us several new sponsors this year - Chaco Canyon Consulting, Microsoft Technology Centers (MTCs), The MathWorks, and most recently AccuRev, Inc. Jim Withall, our Membership Chair, makes sure that all our SPIN members are notified of the meetings. John Desmond, our SPIN Plus Chair, maintains our SPIN Plus distributions, notifying folks of process related information. P.J. Gardner, our Web Designer, has designed a great looking Boston SPIN Web site. Thanks to the other members of our SPIN Steering Committee: Jacqueline Luciano, our new Treasurer; Carol Perletz, our new Secretary; and Dave Heimann, our other Member at Large. And last but not least, our *In the SPIN* editor, Judi Brodman, who has had a variety of roles with the Boston SPIN as well as being a founding member and the SPIN Doctor. Thanks to Judi you're now reading another wonderful edition of our newsletter.

We're off to a great start in our 2004-2005 SPIN year. We hope you'll join us at one of our thought-provoking meetings, a stimulating roundtable, by networking with other SPINners, and/or by reading our Web site.

Boston@SPIN Established January 1993
Software Process Improvement Network

IN THIS ISSUE . . .

From the Editor.....	1
Letter from the Chair.....	1
Contributor Spotlight.....	2
Roundtable Synopsis.....	5
Dear SPIN Doctor.....	7
SPIN Information.....	7
Upcoming Meetings.....	8

Contributor Spotlight

Estimate Remaining Software Defects with TSPSM

Rob Tonneberger, SEI certified Personal Software ProcessSM (PSPSM) Instructor and Team Software ProcessSM (TSPSM) Coach

This article describes the application of a statistical technique called Capture-Recapture (C-R). C-R is practiced in the SEI's Team Software ProcessSM (TSPSM) to estimate the remaining software defects in a unit of code. As a by-product of code inspections, C-R indicates whether the code meets its quality goals or alternatively, should be re-inspected or rewritten.

Fish Populations



What do fish and software defects have in common?

Answer: Their populations can both be estimated using a technique called

Capture-Recapture.

One hundred years ago a Danish fisheries scientist, C.G.J. Petersen, developed a statistical method to estimate fish populations within a lake. Petersen's process of tagging captured fish led to his publishing the method in 1896 [Petersen]. F.C. Lincoln used a similar technique to estimate North American waterfowl populations in the 1930s [Lincoln].

Petersen's technique was to "capture" a number of fish and place a small brass tag on each of them before returning them to the lake. A few days later when the tagged fish had distributed themselves throughout the lake, he "recaptured" a number of fish and counted those having the brass tag. This method is now named the Petersen Estimation or the Lincoln-Petersen Model.

Estimating the total fish population is based upon the principle that the proportion of fish tagged in the recapture equals the proportion of tagged fish in the population as a whole, so that:

$$a / b = c / d$$

where:

- a** is the number captured, tagged and released into the population
- b** is the size of the population as a whole
- c** is the number recaptured in the second catch with a brass tag
- d** is the size of the recapture catch

Rearranging the equation to solve for the total population is:

$$b = a * d / c$$

There are three assumptions required for this method:

1. No immigration, emigration, births or deaths between the capture and recapture.

2. The probability of being caught is equal for all fish, including those tagged.

3. Tags are not lost and are always recognizable.

Estimating fish populations is very much like estimating the total number of defects in a unit of software. Both can use a sampling technique that is proportional to the total population of fish, or in our case, software defects.

Code Inspections

Inspections could be considered the proverbial "silver bullet" in reducing software defects – they are relatively inexpensive yet produce dramatic defect reductions. A code inspection is where people other than its author examine the code with the specific intent of finding errors in it. Michael Fagan formalized and introduced the software inspection process while at IBM in the 1970s, hence the term "Fagan inspection"¹ [Fagan].

Although code inspections have undisputedly proven their value over almost 30 years, most software is still developed without taking advantage of them. The importance of code inspections becomes clear when you consider the following dramatic statistics.

- A combination of design and code inspections usually removes 60 to 90% of the defects in a product [Fagan].
- An AT&T study of an organization with more than 200 people found inspections led to a 14% increase in productivity and a 90% decrease in defects [Fowler].
- HP found that 80% of the errors detected during inspections were unlikely to be caught by testing [Ganssle].
- Jet Propulsion Laboratory estimates it saves \$25,000 per inspection [Bush and Kelly].
- A combination of testing steps (unit, integration, system, etc.) often finds less than 60% of the errors present [Jones].
- Projects that used reviews and inspections had a tenfold reduction of defects found in test and a 50% to 80% reduction in test costs, including the cost of the reviews and inspections [Freedman].

The final and most compelling case for inspections is that five hours or more are saved during debugging and testing for every hour invested in up-front inspections. All of this can be had for a typical investment of only 15% of the project's total resources [Gilb 1988], [Humphrey].

A code inspection uses a team of individuals with roles:

- Moderator – leads inspection process, distributes materials, follows-up on rework.
- Scribe – notes each anomaly/defect on a standard form.
- Reviewer – anyone reviewing the code except the author.
- Author – illuminates unclear areas.

In essence, the process starts with the author submitting his cleanly compiled code to the moderator. The moderator distributes the code and supporting materials to the inspection team. An overview meeting is held if the team is unfamiliar

¹ This process goes by other names such as formal inspection, code walkthrough and peer review.

with the code's function. The inspection team individually examines the code using checklists that insure all potential problem areas are covered. Thereafter, the entire team meets to review the code. The reader paraphrases the code's functionality in small sections at a time. Team members point out anomalies and defects while the recorder notes each and its class as to major (changes the source code) or as minor (spelling, style, workmanship). The moderator maintains a swift pace by allowing only code anomalies to be pointed out, not their resolution. Afterwards, the author makes the necessary changes and submits the revised code to the moderator for satisfaction.

Here are some important considerations in performing code inspections.

- The code compiled cleanly without any warnings and errors.
- The code has had no unit testing.
- The code is individually inspected at a rate not greater than 200 LOC²/hour.
- The inspection checklists are tailored to problems found in the past.
- The code complies to a defined software standard.
- Management does not participate.

"Alright", you say, "I see the advantage of code inspections and I understand how to count fish, but what about estimating the quantity of defects?"

Two-Person Inspection

Let's consider a code inspection having only two reviewers, Sue and John, where they double-up on the inspection roles. Sue, acting as the Scribe, recorded the following data.

* Shaded cells are calculated.

Reviewer	Defects		Preparation			* Est. Yield
	Major	Minor	Size	Time	* Rate	
Sue (reviewer A in next table)	7	**	235	92	153	58.3
John (reviewer B in next table)	5		235	54	261	41.7
TOTALS			470	146	193	75.0

** Minor defects are removed for clarity.



For each reviewer, the table above shows the number of major and minor defects found, the size of the code in LOC and the number of minutes consumed in their review. Rate is the LOC inspected per hour. Although John is a fine reviewer, he rushed this review and found correspondingly fewer defects than usual.

² LOC or Lines Of source Code not including white space or comment lines.

Line	Defect Description	Defects		Major Defects	
		Major	Minor	A	B
10	Initialize rx_flag	1	**	1	
23	"=" s/b "=="	1		1	1
54	NULL the pointer	1		1	
61	CHAR_CNT = 100	1			1
78	">" s/b ">="	1		1	1
79	Check parity	1		1	
112	Use Get_Char instead	1		1	1
138	Misspelling	1		1	
182	Add else clause	1			1
Total Found		9		7	5

** Minor defects are removed for clarity.

INSPECTION SUMMARY

Meeting Time:	<u>43</u>
Product Size:	<u>235</u> LOC
Total Inspection Hours:	<u>4.6</u>
Defects per hour:	<u>1.96</u>
Hours per defect:	<u>0.51</u>
Defects Found for A:	<u>7</u>
Defects Found for B:	<u>5</u>
Common Defects (C):	<u>3</u>
Total Est. Defects (AB/C):	<u>12</u>
Number Found (A+B-C):	<u>9</u>
Estimated Number Left:	<u>3</u>

Here is where the fish story comes into play. The table above shows those defects in detail that Sue and John found. Sue (A) found 7 defects with 3 shared (C) with John. John (B) found 5 defects with again 3 shared (C) with Sue. According to the fish formula, $b = a * d / c$, the total defects are calculated with Sue's major defects (7) multiplied by John's major defects (5) divided by those in common (3) or 11.67 rounded up to 12. With having already found 9 defects, there are an estimated 3 defects remaining. These defects can be found by a re-inspection, or with any luck, through testing. Regardless, at least the team has a rough idea of the defects still in that code and can therefore make informed decisions on the form of remedial action.

Some other useful data is that Sue had an inspection yield (defects found / total estimated defects) of 58.3%, but John only obtained a yield of 41.7%. The total inspection time was 4.6 hours (43 minutes for the meeting times 3 developers (Sue, John and the author) plus Sue's 92 review minutes and John's 54 minutes). The code inspection cost 0.51 man-hours per defect (4.6 / 9) – clearly less than the 2 to 20 hours of testing to find and fix a defect that is typical of industry [Humphrey].

Three-Person Inspection

The development team decided that the least expensive way to eliminate the remaining 3 defects was to have Steve inspect the same code. For an inspection team of three or more reviewers, the method is the similar except that it has an additional step, that of deciding which reviewers are to be A and B (see Table A).

Reviewer	Defects		Preparation			Est. Yield
	Major	Minor	Size	Time	Rate	
Sue	7		235	92	153	58.3
John	5		235	54	261	41.7
Steve	9		235	88	160	75.0
TOTALS			705	234	181	91.7

John and Steve's major defects are shown in separate columns in Table B.

Sue added the data from Steve's inspection into the two tables. However in the table below, Sue, John and Steve's major defects are shown in separate columns.

Line	Defect Description	Defects		Major Defects				
		Major	Minor	Sue	John	Steve	* A	* B
10	Initialize rx_flag	1		1		1	1	1
23	"<=" s/b "<="	1		1	1	1	1	1
54	NULL the pointer	1		1				1
61	CHAR_CNT = 100	1			1	1	1	1
78	">" s/b ">="	1		1	1			1
79	Check parity	1		1		1	1	1
112	Use Get_Char instead	1		1	1	1	1	1
138	Misspelling	1		1		1	1	1
155	Add (...) in calc	1				1	1	
182	Add else clause	1			1	1	1	1
197	"<=" s/b "<="	1				1	1	
Total Found		11		7	5	9	9	9

Shaded reviewers A and B are determined in the text below.

INSPECTION SUMMARY

Meeting Time:	<u>43</u>
Product Size:	<u>235</u> LOC
Number Found (A+B-C):	<u>11</u>
Defects Found for B:	<u>9</u>
Total Inspection hours	<u>6.0</u>
Defects per hour:	<u>1.83</u> Defects
Hours per defect:	<u>0.54</u>
Defects Found for A:	<u>9</u>
Defects found for B:	<u>9</u>
Common Defects (C):	<u>7</u> Total Est. Defects
(AB/C):	<u>12</u>
Estimated Number Left:	<u>1</u>

For column A, find the reviewer that found the greatest number of defects that no one else found. Obviously, Steve found the most defects so put his data into the A column. Next, combine the data from Sue and John and treat the combination as reviewer B by putting a 1 in column B for every defect that either Sue or John found. The total defect is then $9 * 9 / 7$ or 11.57 rounded up to 12. With an estimated 12 total defects but already having found 11 of them, there is perhaps only 1 defect remaining. So, adding another reviewer increased the overall inspection yield from 75% to 91.7% yet the defects per man-hour cost stayed relatively the same – more bang for the buck!

One caution on using the capture-recapture technique - remember the three assumptions for estimating fish. Although our defects don't emigrate and our tags aren't lost, the probability of finding the defect may vary between reviewers. This is particularly true when reviewers are intentionally looking at different aspects of the code and if so, this technique may produce questionable results

References

Bush, Marilyn and John Kelly, "The Jet Propulsion Laboratory's Experience with Formal Inspections", *Proceedings of the Fourteenth Annual Software Engineering Workshop, November 29*, Goddard Space Flight Center. Document SEL-89-007, 1989.

Fagan, Michael E., "Design and Code Inspections to Reduce Errors in Program Development", *IBM Systems Journal* 15, no. 3, 182-211, 1976.

Fowler, Priscilla J., "In-Process Inspections of Work Products at AT&T", *AT&T Technical Journal*, March/April, 102-12, 1986.

Freedman, D.P. and G.M. Weinberg, *Handbook of Walk-throughs, Inspections, and Technical Reviews: Evaluation Programs, Projects and Products, Third Edition*, Little, Brown and Company, 1982.

Ganssle, Jack, *The Art of Designing Embedded Systems*, Butterworth-Heinemann, 2000.

Gilb, Tom, *Principles of Software Engineering management*, Addison-Wesley, 1988.

Gilb, Tom and Dorothy Graham, *Software Inspection*, TJ Press, 1993.

Humphrey, Watts S., *A Discipline for Software Engineering*, Addison-Wesley, 1995.

Lincoln, F.C., "Calculating Waterfowl Abundance on the Basis of Banding Returns", United States Department of Agriculture Circular, No. 118, 1930.

Jones, Capers, *Programming Productivity*, McGraw-Hill, 1986.

Petersen, C.G.J., "The yearly immigration of young plaice into the Limfjord from the German Sea", Report of Danish Biological Station 6, 1-48. Copenhagen, Denmark, 1896.

Wheeler, Bill Bryczynski and Reginal Meeson, *Software Inspection – An Industry Best Practice*, IEEE Computer Society Press, 1996.

® Capability Maturity Model, CMM and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
 SM SEI, Personal Software Process, PSP, Team Software Process and TSP are service marks of the Software Engineering Institute, Carnegie Mellon University.

Copyright 2004, Northern Horizons Incorporated, Brookline, NH 03033

Roundtable Synopsis

November 16, 2004

Techniques and Problems in Project Estimation

Facilitator: Dolores McCarthy, Quality Manager;
Computer Sciences Corporation;
Boston SPIN Roundtable Chair



The group at this Roundtable was interested in knowing about techniques for project estimation and development of better cost and schedule estimates.

Several estimation methods were offered:

1) using the knowledge and experience gained by seasoned practitioners from similar projects in the past, 2) knowing the requirements, 3) using metrics collected from past projects, 4) the Delphi approach using domain experts and experienced facilitators, and 5) using function points for the scope of work, which calls for training to be successful.

Familiar problems encountered in meeting cost or schedule, caused by the estimation process, or lack of it, were presented as were issues related to management, such as difficulty in convincing senior management of the validity of the estimates, management yielding to market pressure to provide estimates too early, being bound by a corporate mindset, or being held to estimates with no flexibility to change with the circumstances.

Other causes of estimation problems could be the data used, for example, using mean estimate data with no variability factored in, using scaling that is nonlinear, or having no historical data because of innovation, that is, attempting something new. Failure to perform risk analysis and failure to compensate for risks were other factors mentioned. A risk example would be vendor dependency – the vendor can't deliver as promised.

Misunderstanding dependencies of parts of the project, or changing, misunderstood, and unclear requirements could all have a negative impact on good estimation. Problems with estimation cause misallocation of resources, cost overruns, and missed milestones.

Participants offered ideas on what could be done to help overcome some of these shortcomings of estimation: 1) Negotiate with the customer to set expectations, 2) Pay attention to customer needs, 3) Revisit the initial estimate after more requirements analysis, 4) Emphasize communication between and among departments and teams, 5) Establish contingencies for risks that materialize, which may impact the cost and schedule, 6) Try phase estimates, e.g., quarters, instead of years, 7) Make the estimate work-product based to gain reliability and Earned Value, 8) Use methods such as a Mini RUP (Rational) or Business Process Modeling (BPM), where

the modeling produces executables, and 9) Let stakeholders set priority and realize a Use Case from the output.

Feedback from the Roundtables showed that more in-depth information is desired concerning solutions to these estimation problems, more focus on one or two methods of estimation, and the pluses and minuses of various project management tools, based on experience with them.

December 14, 2004

Transitioning from the SW-CMM® to the CMMI®

Co-Facilitators:

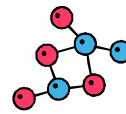
Judi Brodman, Process Consultant, LOGOS International, Inc.

Editor of the Boston SPIN's Newsletter, In-the-SPIN;

Nancy Van Schooenderwoert, Consultant, Agile Rules

The SPIN Roundtable on "Transitioning to CMMI" was very well attended. The facilitators entertained the following questions from the attendees:

What is the difference between SW-CMM and CMMI? Is



the CMMI more prescriptive?

The key difference between SW-CMM (Software Capability Maturity Model) and CMMI (CMM Integration) is that the SW-CMM addresses managing and developing software, while the CMMI addresses more than software - it also addresses systems engineering and acquisition.

Is the CMMI more prescriptive than SW-CMM?

The SW-CMM and CMMI have very similar KPAs (Key Process Areas) Level 2 except for the addition of the new KPA in the CMMI, Measurement and Analysis. Level 3 changes dramatically from the SW-CMM to CMMI – the SW-CMM has 7 KPAs; the CMMI has 14 KPAs. Software Product Engineering, a single KPA in the SW-CMM, is broken down into a number of KPAs in the CMMI and addresses the development practices in much more detail.

Which representation of the model: Staged vs. Continuous?

SW-CMM and/or CMMI maturity rating applies when an organization is appraised at a given maturity level – Level 2, 3, 4, or 5. Each level, in both cases, SW-CMM or CMMI, is defined by a set of key process areas. To receive a Level 3 maturity rating, the organization must have satisfied all the goals pertaining to all the KPAs for Levels 2 and 3. If the organization chooses this structured approach to Software Process Improvement (SPI), it uses the "staged" representation of the model. On the other hand, if an organization chooses to improve certain areas that may be causing them problems, such as Project Management, then the "Continuous" representation would be chosen. In other words, they are choosing which process areas are important to them and their business goals, and implementing the related KPAs in those areas. This way they can attain a higher capability level in the selected process area(s).

What is the recommended path to meeting the goals of the CMMI?

Judi recommended that organizations, without any real process knowledge, who are interested in implementing the CMMI first look into implementing SW-CMM - the reason being that the CMMI is quite a big step up from the CMM and if you can master the SW-CMM, the transition to the CMMI is much easier. Also, along the way, the organization may decide that SW-CMM alone is achieving all the SPI that they needed.

A participant noted that since the CMM will no longer be supported after 2005 by the SEI, maybe CMMI is just an attempt to force companies to buy more training and consulting, etc. There isn't a good answer to why the SEI is unwilling to let the SW-CMM just continue along for those who want just software SPI, and not the full CMMI. Nancy mentioned that with Agile practices (like Scrum, XP, etc) the balance between quality and cost is governed by the project sponsors, but there does not seem to be a mechanism to counterbalance the tendency for the SEI to get more and more stringent in the control they exercise on companies via CMMI.

How does CMMI compare to Agile methodologies?

Judi said that the SW-CMM and CMMI are more focused on what has to be done and, to a lesser degree, how it is done while XP (for example) gives specific practices (the how) for the software developers to follow when writing code. Nancy agreed that XP does mainly focus on software practices, but it can work with other management practices such as Scrum or the SW-CMM/CMMI. Nancy and Judi both agreed that it should be possible to use an Agile software development method within CMMI structure.

The attendees and facilitators agreed that they had just touched the tip of the iceberg and continued Roundtables on these subjects in the future should be setup.

“Meetings, Meetings, Interminable Meetings! How Can We Make Them Less Onerous and More Productive?”

Facilitator: Rick Brenner, Chaco Canyon Consulting

Boston SPIN Vice Chair

The following six ideas on meetings were discussed at the Roundtable:



* When you send out a proposed agenda, make two columns. The left column is a list of proposed issues for discussion. The right column is a list of items that will not be discussed at this meeting. Having a not-agenda helps keep the meeting focused.

* Beware of “cafe” format. Sometimes a meeting erupts into cafe format -- multi-threaded, simultaneous discussions of possibly unrelated topics. This is sometimes a signal that the group needs some social time, and it happens more often

when the group assembles only for business meetings, and never any other purpose. To prevent cafe eruptions, allow time for socializing -- an occasional lunch get-together, or a breakfast, lunch, or pre-meeting refreshments. Don't mix the linear, business format with the cafe format.

* Send reading material by email in advance Give people enough time to read and digest the material before the meeting. Springing it on them at the meeting is a distraction and often counter-productive.

* Have a Parking Lot but beware of Parking Lot Abuse. Keep a “Parking Lot” on a whiteboard or flip chart. It's a list of items that come up in discussion, but weren't on the agenda. To keep from forgetting them, and to make it easier to suspend discussion of them, park them in the parking lot, and promise to take them up later. But beware of abusing the parking lot -- that is, parking something with no intention of ever discussing it. If you park something, it's a promise to discuss it soon.

* “We're not here to discuss. We're here to resolve.” Discussion is important, but only if it leads to resolution. Maybe not in one meeting, but eventually. Restrict the discussion to content that will help the group come to a resolution about some specified issue.

* Have a Designated Digression Detector. Appoint someone to notice digressions, and then alert the meeting to them. Then participants can decide whether to suspend the discussion, “park it” or keep going.

“How Do You Reward/Motivate People When the Budget Has Been Cut?”

Facilitator: Chris Reeve, Stratus Technologies



This roundtable talked about various methods of rewarding and motivating people when there is little money to be spent due to budget cuts.

First we discussed various rewards for project completion. Some of the ideas mentioned were: silly gifts, T-shirts and impromptu celebrations. Along with these tangible items, emails to the rest of the company informing them of the team's accomplishments were recommended. Someone also suggested they had used a prominent “employee of the month” parking space as a reward. When outside purchases are being limited, someone suggested asking the Marketing department if they had extra promotional items for a giveaway, or raffling off the company stock, or coupons to the company's cafeteria.

One person related a story of a company they'd worked in where the manufacturing staff was allowed to leave early daily if they had completed their assignments.

The group also spent time talking about motivating employees under these circumstances. Some of the suggestions were: taking away the interruptions from people's jobs, eliminating

the hassles of the job, offering a career path, starting a mentoring program, starting an internal training program and letting people know they are on the “keep list” when there is an impending layoff.

The facilitator also asked people what was the best reward they received, and what they recommend NOT be implemented. The best rewards were: the celebration after the project, a \$1000 bonus, a T-shirt, and a week in Germany. It was recommended to avoid: silly gifts as they can sometimes backfire, inequity in award distribution and not to give visible rewards to managers in lieu of rewards for the troops.

The participants provided for a lively discussion and a variety of ideas were expressed.

Dear SPIN Doctor

.Dear SPINners:



It's a new year - let's fire-up this column again. Send any questions you have on any software areas and, if the SPIN Doctor can't answer them, she will find an expert who can! Review some of the old columns and see who the guest columnists were!

Remember, this column is for you; let's make a difference! Send your comments and questions to “Dear SPIN doctor” at: <mailto:brodman@logos-intl.com>. Sign them or use a “pen-name” -- I respect your confidentiality!!

-- The SPIN Doctor

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.boston-spin.org>.

For more information about our programs and events contact Donna Johnson, Program Chair, johnson@logos-intl.com.



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.

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 ad-

mission 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.boston-spin.org> or on The MITRE Corporation Web site: <http://www.mitre.org>.

Our Sponsors



Our appreciation goes to all our generous sponsors for supporting our SPIN activities!!

Chaco Canyon Consulting

<http://www.ChacoCanyon.com>

Chaco Canyon Consulting works with people in problem-solving organizations who make complex products or sophisticated services that need state-of-the-art teamwork, and with organizations that want to achieve high performance by building stronger relationships among their people.

AccuRev, Inc.

<http://www.AccuRev.com>

AccuRev provides award-winning, stream-based software configuration management tools that provide both the flexibility and built-in best practices that developers need to manage today's complex, parallel, and distributed software development projects.

The MITRE Corporation

Microsoft Technology Centers (MTCs)

The MathWorks

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

Email Lists

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

Jim Withall, Boston SPIN membership chair, at JWithall@OneBeacon.com.

To receive Boston SPIN-Plus specific notices, send an email to: spin-plus-request@boston-spin.org, include “subscribe” in the message body.

To post an announcement on the SPIN-Plus list: spin-plus@boston-spin.org

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:

- Judi Brodman, 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.boston-spin.org/>.

Upcoming Meetings

Date:	Speaker:	Topic:
1/19/2005 Joint SPIN/ASQ meeting	Unmesh Gundewar	Build the “Right Software” to Delight Your Customer - A Quantitative View
2/15/2005	Michael Mah	Agile Management Methodologies
3/15/2005	Peter Hennessey	A culture of discipline with an ethic of entrepreneurship; can ISO, CMMI, and Agile methods co-exist?
4/19/2005	Barry Mirrer	Organizing Your Problem Tracking System
5/17/2005	Steve Rakitin	Building Accurate Schedules from Software Requirements
6/21/2005	Johanna Rothman	Successful Software Management: 15 Lessons Learned

Enjoy our New England Winter!!!

