A Problem-Based Approach to Software Process Improvement: A Case Study

Johanna Rothman
Rothman Consulting Group, Inc.

Organizations struggle [1] with their process improvement efforts for a variety of reasons. Perhaps the most common struggle pattern is to take a long time developing a general understanding of their processes and then trying to define all possible alternatives in the product development process. This pattern leads to large, unmanageable, unreadable, and incomplete [2] process documentation.

This paper is a case study of one organization that minimized the struggle by taking a different approach to the development of their product development process.

Introduction

The organization described in this paper was a 12-year-old company, formed out of two startups. It created and sold graphics products. We will call this organization "ExtendIt." ExtendIt employed about 150 people worldwide. The product development staff was split into two locations: about 50 people in the Boston area office, and about 20 people in a European office.

ExtendIt was in a typical chaotic state — most of senior management did not understand the software product development process. Engineering management did not know where or how to begin, project management and product management were nonexistent, and engineering processes were completely inadequate for product development and testing. Projects were planned for four to eight months, but typically took 13-18 months. Even at the end of the extended development time, ship decisions were generally based on emotional reasons to ship, not objective reasons. For example, management made the decision to ship a major release because the developers were too tired to continue the 80-hour weeks, not because the project met the ship criteria. In fact, that particular project did not have all the expected features, so the developers continued to work long hours to get the features into the follow-on release.

Approach to Process Improvement

A new CEO started at ExtendIt and changed the product strategic vision and sales model. Based on the new goals, it was clear that the organization had to change how it developed products. It was not possible for this geographically dispersed engineering organization to meet the new goals without changing their practices.

Senior management had already agreed to decouple releases from project development, which is a typical concurrent engineering approach to product development. This would be known as the "release train," a quarterly plan to ship products[1]. Projects at a certain point in their development would be eligible to be loaded on the train and be shipped. Projects would not be shipped unless ready. To meet the release train goals, ExtendIt formed small independent projects.

A software engineering process group (SEPG) [3] was formed in May 1997, with the original plan that the process definition and design could be completed by the end of July 1997, a total of eight weeks. The SEPG consisted of engineering management such as the vice president (VP) of engineering, the documentation manager, development, and release engineering managers; the director of program management, and an outside consultant — a total of seven people. The initial roles of the people on the SEPG were:

- The VP of engineering was the facilitator between the SEPG and organization at large.
- The documentation manager served as the chairman of the SEPG and provided expertise about documentation processes.
- The two development managers and the release engineering manager provided expertise about current processes and how they could be changed.
- The program management director provided specific engineering expertise and general organizational expertise about product development.
- The consultant provided planning and facilitation for the SEPG meetings in addition to process and product development expertise during the process design.

Like many organizations, the SEPG planned to roll out the process definition and templates to the organization à la the hole-in-the-floor model of change [5][2]. The rollout milestone was planned for August 1997. After the initial SEPG effort, engineering management was to carry out ongoing process change. This SEPG forgot one thing — change is not successfully rolled out to organizations [5]. People have to integrate the changes into their daily lives for the change to be successful. Although this SEPG did not anticipate this, changes were introduced and integrated into the organization in a most fortunate and successful way.

Problem Statement

The SEPG began by discussing what had to change. Using brainstorming, they identified 29 issues. Then they used affinity grouping to sort the 29 issues into nine "buckets"[3]. Each SEPG member cast three votes, and voted on their top three

issues. They took the top 80 percent of the problems and threw away the lower 20 percent. The result of this analysis were the following six problem statements:

1. The product development process was not documented. The process was not uniform among projects.
2. The functional specs/design specs were not separated. Because the functional description and the design was intertwined, some parts of the system were not well-defined and the test planning effort was insufficient.
3. Vague marketing requirement documents (MRDs) told development how, not what, to do.
4. Development's intake of market requirements were not well defined or controlled. This was really an organizational problem — getting a single point of contact for discussing issues.
5. Too many off-process interruptions. The engineering staff was interrupted or dragged off to work on other issues. There were no organization-wide rules about how to get consulting from others.
6. Managing to a schedule was a problem. People did not know how to manage their own time, or how to rank their activities.

Each SEPG member wrote six descriptive sentences describing each problem as it appeared to or affected each person. The SEPG called this their "6x6" matrix, for six sentences about each of six problems. Everyone's sentences were gathered into a concept matrix, with each major item on the left, and the relevant issues on the right. The SEPG then grouped the problems into subcategories, to organize the issues. (See Table 1 for a representative portion of the final set of problem statements.)

The final concept matrix has a generic problem statement, specific issue, and examples of how each issue affected the organization. The SEPG then made a critical decision — the SEPG decided to focus its work on just the six problem statements above: documenting the product development process; separation of functional and design specs; specific MRDs; how development took in requirements; managing interruptions; and managing to a schedule. This focus provided these main benefits:

- SEPG MODELED PROBLEM-SOLVING BEHAVIOR — Not every decision was correct in hindsight, but the problems were discussed in context of the problems the SEPG was trying to solve. The decisions and the decision-making process were accessible to the organization.
- SEPG PRACTICED PROBLEM-SOLVING SKILLS — The managers were on the SEPG. They had a chance to practice their problem-solving skills in an environment of their peers, before trying them out on a project. This included practice using the traditional problem-solving skills and tools, such as brainstorming, affinity grouping, and facilitating discussions of diverse ideas.
- FOCUSED SEPG WORK — ExtendIt was working towards a rational way of doing business, not towards public certification or assessment. Using the business as incentive for the process improvement activities was understandable by the management and technical staff.

Table 1. *Portion of concept matrix describing problem statements.*

Product Dev. Process not documented	A: Central Reference Required	1. There is no "playbook" which can be given to all employees, so they know the process for developing software. 2. Missing the "what, when and who" for product development maintains our current (perhaps our past) operating procedures (i.e., controlled chaos) 3. Without a documented product development process, there is no real way to determine where we are in the development process (there is no "starts with" or "ends with" statement or entry or exit criteria). 4. …
	B: Common Terminology	5. Terminology is imprecise, e.g. "Alpha" means something, but not the same thing to different people. 6. …
	C: Phase Definition/Criteria	7. There is no current opportunity to define project success criteria. 8. There is no current opportunity to know when a project is complete. 9. There are no clear phases, with entry/exit criteria to know what is done and what is not. 10. …

Intermediate Results

The VP of engineering and some SEPG members felt very strongly that some aspects of product development could not be planned. The VP wanted the SEPG to take an approach to process definition that facilitated reasonable things for reasonable people to do. The SEPG would then incorporate management reviews into the process that were sufficient to inform management, and enable management to take appropriate steps. In addition, the process documentation would give general problem-solving guidance. (Online documents describing useful meeting techniques and project management techniques were part of the final deliverables.)

The SEPG approached the process definition work as if it were an engineering project. The work started with a strawman five-phase process:

- Concept/Requirements
- Design/Definition
- Coding/Implementation
- Validation/Verification
- Manufacturing/Ship

Starting from its charter, the SEPG initially refined its concept (Concept/Requirements phase). The SEPG took the time to define its requirements and an initial project plan, to clarify project completion and success criteria. To clarify and define SEPG deliverables, the initial SEPG project plan used the five phases above.

During the design and definition phase, the SEPG defined the functional specification and design. The SEPG made an initial cut at the phases, figured out what the necessary documents had to be, and where the review points were.

The implementation phase consisted of the detailed design of the process description, and generating the flow charts and words to describe it. To get

early testing, and to get engineering buy-in, the SEPG held focus groups to discuss each phase. Getting early engineering input had these benefits:

1. The SEPG's work was visible to the organization. In fact, parts of the organization were able to test the process by using pieces of it on ongoing projects. Doing this early testing has some ramifications:
   - The SEPG could see if the people who were supposed to use the process would actually use it.
   - A number of issues arose during these focus groups. The discussion around these issues allowed the SEPG to change and simplify the process.
2. The SEPG was able to gain substantial experience in presenting the process to the organization. When the focus group was confused, the SEPG could test how the focus group understood different descriptions.
3. The SEPG walked the talk of "early and often" review and testing. By having their work held up for review and verification, it was easier for the engineering staff to buy into frequent reviews and early testing.
4. Using an evolutionary process design meant the SEPG did not have to get everything right the first time. The engineering organization could see this, and see the relevance to their work.

At the end of the implementation phase, the five-phase product development process had evolved into:

- Concept/Requirements
- Design/Definition
- Coding/Implementation
- Validation/Verification
- Product Qualification

Disadvantages of this Approach

The SEPG worked very quickly, so it was hard for some people to integrate the changes to how they thought. Although the SEPG members did not have trouble with the concept of iteration, some had trouble with their ability to iterate their thoughts quickly. These SEPG members were thrown into chaos [5] with almost every meeting, and had a difficult time adjusting to the pace of change. Change can be painful to the people involved.

During the SEPG's work every member had to closely examine and change or give up closely held ideas about product development. Changing your mind about something when you do not have direct experience with its potential for success can be very hard. Some of the SEPG members were quite reluctant to change how they worked, even when they admitted their current patterns were not working.

For example, the SEPG intellectually understood that inserting a milestone at the beginning of the Coding/Implementation phase to verify the release criteria against each project's criteria made sense to everyone. Some SEPG members were concerned that these release criteria would be fixed too soon and would be nonnegotiable. They were concerned that they would be forced to develop the wrong product. The rest of the SEPG, from experience, realized that clarifying release criteria before the code is finished is one easy way to make sure that the product under development is the correct product. The reluctant SEPG members were concerned because they had no experience with the success of release criteria. They knew their current methods were inadequate, but were reluctant to agree to something they had no direct knowledge of. As an SEPG, we agreed to conduct mini-retrospectives during the first few projects, to check on this and other points in the process.

Some of the SEPG members also had trouble changing their meeting behavior. Some team members were stuck in legacy behavior, using the same assumptions that had created the problems. One assumption was that all decisions were open to more discussion and change after the decision was made. It was impossible to make progress when all decisions could be revisited at any time by anyone. Consequently, the SEPG remained stuck in the "storming phase" of team development [4]. After discussing these problems with the SEPG chairman, the consultant requested the VP of engineering attend some team meetings. The presence of the VP acted as an inhibitor to "business as usual," and allowed the team to make appropriate decisions and move forward. In the case of the SEPG's decision-making, the VP verbalized the SEPG's responsibilities and the time to deliver on those responsibilities.

Results of Using the Process

SEPG Results

The original dates were very aggressive (an eight-week schedule), and were not met. Missing the original dates created these results:

- The SEPG was able to practice iteratively replanning its schedule. This experience was directly applicable to normal engineering projects.
- After the first milestone was missed, the SEPG practiced testing its work focus. Were members working on the most time critical and valuable item?
- The SEPG clarified its tradeoff decisions and decision-making process. It created a "Pending Bin" to place ideas and issues that were relevant to address, but not now.

All these issues emulated typical challenges of a product engineering project. The SEPG gained the understanding that its work was a process development process. The end result was not a saleable product, but it was a process where similar tools and ideas were useful.

Product Development Results

Initially, the engineering staff was concerned about changes to how it was expected to do product development. At the initial overview presentation of the release train, the engineering staff was confused by terminology and how to do what, because the specific changes to the process were not rolled out. The SEPG started its work after this initial presentation.

To get buy-in from the engineering staff, the SEPG started focus groups to discuss the process steps and then the templates in group meetings. The SEPG chose one SEPG member to present each life cycle phase to the focus group. The focus group would ask questions, and the designated SEPG member answered the

Typical problem	Avoidance
Process improvement effort takes a long time.	Focus the SEPG's efforts on the problems that need to be solved now. The engineering staff was not only willing to use the process; it demanded it on projects.
Define all possible steps in the process.	The process provides reasonable guidance and specific criteria for escalation to management. Project participants use the process as a guide. They use their judgement about how to deal with problems, until the problems need to be escalated to management.
"Big Honking Binder" syndrome: the size of the documentation overtakes the process definition.	One-page process flows with one-page descriptions. One- and two-page document templates are part of the process definition. The whole process document is 20 pages.
Technical staff is suspicious of process development process and reluctant to adopt outcome.	Test the process with staff as it is developed.

Table 2. *Lessons learned by ExtendIt.*

questions. The rest of the SEPG staff took notes about the presentation and the questions. When there were many questions, the SEPG generally redesigned the process to make it easier to understand, easier to implement, and more streamlined.

After the process was reviewed in the focus groups, the templates (plans and specification documents) were reviewed in focus groups. The SEPG used the same process: one SEPG member presented the material, and the focus group commented on the material.

By the end of the focus group activity, all the senior staff in engineering had seen parts of the process and the templates. Because the engineering staff helped create and review the process and the templates, the senior staff led the rest of the technical staff to adopt the process. At the next general presentation, the overall process was discussed. The engineering staff understood the process and the templates and it had been made clear what they had to do and when.

Lessons Learned

ExtendIt employees learned a tremendous amount from these steps to process improvement: a process improvement process. They were able to avoid some typical process improvement problems shown in Table 2.

Conclusions

This process improvement process was very effective. It consisted of first determining the problems that needed solving, then developing a process that illustrated the way to do the general case, and a set of problem-solving skills. About eight weeks after the SEPG formation, the SEPG members began to work differently. The SEPG thought about their deliverables to each other in a more complete way, i.e. how people could use what they developed, and the effects of their deliverables on other deliverables.

The biggest organizational change was that the managers and technical staff thought differently about how to do their work. They started to plan for the reasonable case, and created a risk assessment and management plan. This had the desired effects of creating simpler project plans, and pushing risk assessment into the organization.

A small process description seems to be adequate for the present for this organization. The process description contains five pages of flowcharts, about four pages of definitions, and about five pages describing the process and general problem-solving techniques. In addition, there are templates for each document the engineering staff produces.

ExtendIt has been using this process for almost a year. It has successfully produced three quarterly release trains. The technical and management staff has tested the process, and for now, it works.

ExtendIt has had a difficult time escaping from its startup phase. The new CEO and senior management are determined to make the company a success. From a product development perspective, the organization can now deliver products on time and within budget, with the requested features. Using the release train to chunk the features into smaller independent projects, and by creating the expectation that the organization would deliver multiple products over the course of the year, ExtendIt is operationally poised to succeed. ◆

About the Author

Johanna Rothman speaks, writes, and consults on managing high technology product development. She works with her clients to increase the effectiveness of their managers, helping them ship the right product at the right time, and hire and retain the best people. She has more than 20 years experience in software engineering and management and is part of the clinical faculty of The Gordon Institute at Tufts University. Rothman holds two American Society for Quality certifications: Certified Quality Auditor and Certified Software Quality Engineer.

38 Bonad Rd.
Arlington, Mass. 02476
Voice: 781-641-4046
Fax: 781-641-2764
E-mail: jr@jrothman.com
Internet: http://www.jrothman.com

Acknowledgements

In addition to the anonymous reviewers, I thank the following reviewers for their help and substantive comments: Don Gray, Brian Lawrence, Sue McGrath, and Jerry Weinberg.

References

1. Hayes, Will and Dave Zubrow, *Moving on Up: Data and Experience Doing CMM-Based Software Process Improvement*, CMU/SEI-95-TR-008, 1995.
2. Murphy's Law. Specifically, "Whatever can go wrong will, at the worst possible time."
3. Humphrey, Watts, *Managing the Software Process*, Addison-Wesley, Reading, Mass., 1989.
4. Scholtes, Peter R. Joiner and Streibel,

The Team Handbook, Joiner Associates, Madison, Wis., 1996.
5. Weinberg, Gerald, *Quality Software Management: Volume 4: Anticipating Change*, Dorset House Publishing, New York, 1997.

Notes

1. Companies who have the need for parallel development of multiple releases use this concept. Although Sun has implemented this differently, the release train idea described in http://solaris.license.virginia.edu/sun_microsystems/workshop4.2_docs/teamware/solutions_guide/casestudy.doc.html No. 8868 is similar in concept.
2. The hole-in-the-floor model of change: Some set of people upstairs develops the perfect system. The change plan consists of drilling a hole in the floor. The system is dropped through to the people below. Supposedly people instantly change to the new system. Unfortunately, people generally cannot change without integration and practice.
3. Affinity grouping is the activity of creating sets of similar ideas together under one theme. In this case, we wrote each problem on a sticky note, silently organized the sticky notes into groups, and then named each group.

"High Leverage Best Practices..." continued from page 14.

Management. This identifies and describes in detail the 16 Critical Software Best Practices. SPMN, in coordination with the Airlie Council, also developed a related implementation handbook and compact disc, that is in beta testing, and that is being enriched through a review process by a tri-service group of program managers. These materials, along with related briefings, video clips, and other material are available without charge from the Software Program Managers Network at www.spmn.com. ◆

About the Author

Norm Brown is the change agent for more than 200 Army, Navy, Air Force, and other defense programs. He founded the SPMN, which conducts benchmarking of commercial best management and technical practices, and provides technical and management consulting support to these programs, and which has some 10,000 members. He founded the Airlie Software Council, led the Defense Department's Software Acquisition Best Practices Initiative, and was a member of the DoD Software Management Review Board. He chaired the Joint Logistics Commanders Group on Technical Data and Computer Software and the group that drafted the Federal Acquisition regulations on Technical Data and Computer Software, and numerous other groups and committees. He was responsible for the review of more than 300 defense software acquisition projects. He served in numerous acquisition selection boards and advisory positions to the Secretary of the Navy, Deputy Secretary of Defense, and other defense officials. Brown was the software manager for a number of real-time Navy weapons systems and was a commercial software developer.

Software Program Managers Network
P.O. Box 2523
Arlington, Va. 22202
Voice: 703-521-5231
Fax: 703-521-2603
E-mail: spmn@aol.com

References

1. News Release, Office of Assistant Secretary of Defense (Public Affairs), Washington, D.C., July 13, 1994.
2. House Armed Services Committee Report 106-162, pp. 198.
3. Senate Armed Services Committee Report 106-50, pp. 340-341, May 17, 1999.

Note

1. The Software Engineering Institute has developed the Capability Maturity Model, which identifies where a company lies along a continuum of software development maturity.