CUTTER IT JOURNAL
The Journal of Information Technology Management

CUTTER CONSORTIUM
ACCESS TO THE EXPERTS

February 2003 Vol. 16, No. 2

XP and Culture Change: Part II

Bold Leap Forward
Getting the business to accept responsibility and authority for the scope of systems (the major organizational implication of Extreme Programming) is reasonable and possible, and the results are satisfying and valuable.

Mission Impossible
Organizations don't want to change, and organizational suggestions from technologists will simply be ignored, no matter the scale of problems or the opportunities if those problems are solved.

"We have often failed to communicate the possibilities inherent in what teams can now accomplish in terms non-geeks can understand. "

— Kent Beck, Guest Editor

opinion

It Gets Worse Before It Gets Better:
Changing to XP

by Lowell Lindstrom and Kent Beck

A FABLE OF CHANGE

It's the late 1990s. Amazon.com has just done its hockey stick on the stock market. The gold rush is on. Internet stocks go through the roof; that pulls up telecom stocks, and IT consultants hang up a new shingle and see their valuations soar. This is a party not to be missed. You need to change.

You have also been busy ending the millennium. You have invested lots of time and money trying to improve your software development capabilities. You've reduced defects, begun to deliver in shorter cycles, and have greatly improved your project management skills. Requirements changes and bugs still wreak havoc and often lead to the feared death march, where all discipline is abandoned in order to meet the deadline. All in all, though, you are doing well.

Now these hot-shot Internet programmers have changed all the rules. They have no legacy code or installed base of customers. They have these cool new languages and platforms with which to develop. No one seems to care what process or project management approach they use. Their customers seem willing to test the product for them. And investors are screaming at them to spend more money. Go figure!

Then one of your developers hears about this new XP. Ta-daaaa! The cavalry of practices has arrived.

It is an inflection point, a paradigm shift, a disruption, a chasm-crossing tornado, and now it's in your lap. The rallying cry goes out, "MAKE THE SOFTWARE INTERNET READY … oh, and don't forget to keep our existing customers happy, too." Your journey begins. Some of your developers have been screaming for this, and they are thrilled. They'll finally have some bragging rights at the local e-café as they quaff their Red Bulls. The business strategists are thrilled that you'll be churning stuff out in Internet time. No more waiting on the developers.

If only everyone was so sure about this. Many are still skeptical. There were good reasons to add the processes over the last few years. This code-fix mentality was abandoned years ago. "Sure, you can sell books over the Internet, but our business is different." Is this just a knee-jerk reaction on the part of management? The team has been there. Is this just the trend of the month? There is only one way to find out.

To keep things from getting out of control, you start by turning up the heat on the existing methodology. There must be efficiencies waiting to be excavated. Performance gets worse. You're not delivering fast enough, and the team is falling apart. Chaos ensues as folks break away to work all different ways. You bring in expensive "e-consultants." They get rich as you fail. They convince your management to outsource some of the project to them. More money spent, still no results. Your team is demoralized. Developers are jumping ship, but the pressure remains.

GOOD NEWS

Then one of your developers hears about this new Extreme Programming (XP) [1]. Ta-daaaa! The cavalry of practices has arrived, the longed-for solution. Sure, the team takes a while to integrate the practices of XP into daily life, but things rapidly get better, bumps aside. Once XP is embraced by the whole organization, life is good. The Internet becomes an opportunity instead of a threat. The collapse comes, but you survive because you can adapt to your customers. Cue music, fade to black.

Our experience in introducing teams to XP is that it causes enormous pain and dislocation.

BAD NEWS

Pretty story, isn't it? Total crap. Our experience in introducing teams to XP is that it causes enormous pain and dislocation. Many organizations' change antibodies successfully repel the infection. People still get unhappy, leave, change jobs. Teams still have problems, deliver iterations with no completed stories, experience stress. What's going on? Why isn't our XP fairy tale coming true?

You Forgot to Give Me My Change

To understand this better and to manage it, we must explore how organizations change. That is what we are trying to do, after all. The study of organizational change is not new, yet most organizations struggle to transition to new ways of working, even when they are aware of the need for change. What we need in our toolkit is an understanding of the best practices for changing the way software developers work. Gerald Weinberg [5] discusses a number of change models. He presents, as the most descriptive model, the Satir Change Model [3] (see Figure 1) and finds "the insights from this model essential to the successful transformation of software organizations to cultures that are capable of producing higher-quality software, cheaper and faster" [5].

1. The **old status quo** (OSQ) is the old way of working. The **foreign element** intrudes, and the team can't cope using the old way of working.
2. During **resistance**, the team denies there is a problem or says there is no possible solution. Eventually the team breaks through.
3. The reality check triggers **chaos**. "Houston, we have a problem." Now what? A **transforming idea** surfaces that might help.
4. The team puts the transforming idea to work during **integration**.
5. During **practice**, things continually get better, with occasional bumps.
6. The team reaches a **new status quo**.

To understand the model, we can review the reaction to the Internet in our story above.

The old status quo is the software technology environment of the last decade. Client-server transitions are popular; Y2K fears consume the IT manager. Our foreign element, the Internet, is emerging as a business backbone, but the full extent of its impact is unknown. As the reality becomes clearer, resistance is common. The available data and information indicate the need for change. Still, teams stick with their current, comfortable practices, and performance does not change much. Finally, the need to change is embraced. Management sends down the edict (er ... goal). Whew! Now you can get down to "e-business."

Not so fast. This Internet stuff is not as easy as we thought. Not the technology, not the business model. The team thrashes and has second thoughts. Welcome, chaos. This stage is characterized by wide fluctuations in productivity, an increase in the stress and discord on the team, and second thoughts about this new initiative. The ship has left the dock, it's not going back without a mutiny, and it's too far to

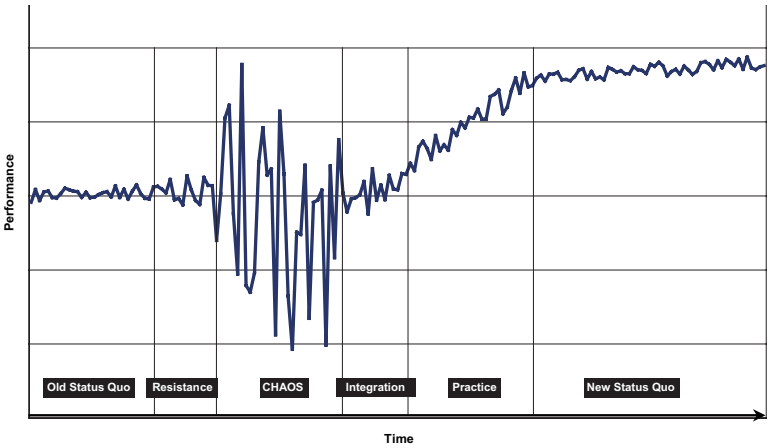

Figure 1 — The Satir Change Model [3].

swim back. In Satir's model, chaos is an expected, inevitable stage of change through which we must persevere. No chaos, no change. Unfortunately, it is rare for teams and organizations to prepare for this period. This lack of preparation worsens an already unstable situation. The biggest mistake is to not expect and be ready for chaos.

To understand the chaos stage, one can usually look to personal experiences. Adjusting to a new job, manager, or team member often causes similar chaos for an individual or team. Our personal lives also include these chaos periods (in fact, the model was developed around families coping with change). Changes to your golf swing or learning a new craft involves this period of awkwardness. But eventually, with pertinacity, what was clumsy becomes comfortable.

We emerge into a period of integration, when we can finally put the pieces together and see how the change enables better things in the future. Integration is triggered by some transforming idea, the "aha!" when we finally understand how to succeed using the new practices. This might occur when we get some confirming feedback on the new Internet business model or we understand how to treat the Internet as an opportunity rather than a threat.

We're not good yet though — that comes through repetition and practice. The thought of the old status quo, however, no longer provides any more comfort than we currently experience. Practice simply makes us better as we approach a new status quo at a higher level of performance. The change is finally paying dividends.

Are We Done Yet?

That's not so bad; all we have to do is manage some chaos for a little while. As a manager, you can meet with the team, describe what is about to happen, open up communication channels. Tell them it will hurt for a while, but then it will get better. XP is going to be our transforming idea; we'll simply streamline that chaos stage by using XP.

If only it were that easy. What we are observing as teams move to XP is the Satir Change Model at a different level in the organization. The old status quo is the old way of developing software (including the old relations between business, development, and management). We've turned the story on its head. In addition to being our solution and transforming idea, XP is now the foreign element. How will the team cope?

OBSERVING XP TRANSITIONS WITH THE SATIR MODEL

As with most advances, Extreme Programming was initially met with broad resistance by much of the industry. As XP found its initial group of innovators and early adopters, more and more teams set out aggressively to implement XP within their teams and their organizations. More recently, we have begun to see organization-wide deployment of XP involving hundreds or thousands of developers and their customers.

The organizations we have aided in this transition have all exhibited the behavior predicted by the Satir Change Model, for good or for bad. To add to the complexity, it is not as simple as one team going through one XP transition. Rather, the transition is going on at different levels and at different rates and for each of the XP practices. Individuals adapt to change at different rates. Change is taking place at four levels of practice: individuals, teams, organizations, and industry. Each practice may exhibit different characteristics of the model for different individuals and teams. Yet despite the complexity of the change, many XP transitions succeed across a broader set of domains and in teams larger than originally thought possible.

Old Status Quo and the Foreign Element

There really is not a particular pattern of OSQ that sets the stage for a team's transition to XP. There must at least be a belief among some influential people in the organization that the OSQ will not meet the needs of upcoming projects. The developers may be seeking the opportunity to take control of the way they work. The customers may be looking for more predictability and the ability to steer the project. Teams that face quality problems may be attracted by the testing emphasis of XP. We have even coached one team whose management was attracted by XP's

sustainable pace, as they had just completed a rough death-march project. Our tale above describes one, perhaps already dated, motivation for XP. Unfortunately, we also consistently see that individuals do not know what the foreign element is or why they are being asked to leave the OSQ in the first place.

Resistance

Resistance to change is a natural human reaction. People resist change for a number of reasons, and resistance takes many forms. Here are some examples we have seen:

- "We do XP, you bet we do. Everything but unit testing. We can't possibly unit test our software. Well, unit testing and onsite customer. We can't get an onsite customer. That and … "
- A senior project manager, unsure of his new role with XP, spends hours in an office with the tracker reanalyzing data that needed no analysis to begin with.
- Previous team leaders take a passive role, doing the minimum they can do without being considered disruptive.
- Lowell had a client in the customer role that unconsciously arranged four 4x6 cards into what resembled a sheet of paper and proceeded to write in, as small as possible, all of the details that she knew the developers needed, even though they had already estimated the story and sketched the acceptance tests.

We *need* to see XP as a solution, so we have a hard time when at first it appears only as a problem.

- The team leads on a project did not know how to succeed without dictating tasks to the developers, even as the rest of the team rose to the challenge. Although the leads professed support for the change, informally they were spreading dissenting interpretations of the day's events.
- The XP transition becomes a lightning rod in the organization for anything bad that happens. Problems that have nothing to do with the development of software are blamed on the XP transition.
- XP can't work because it violates the laws of physics/economics/psychology/sociology/business. Kent was once told that German programmers couldn't possibly do XP because "German programmers are different." Further probing showed that the difference was elusive enough that it was unexplainable to a mere American....

Anyone who has tried to change an organization has his or her own list beyond this small sampling of ours. Resistance is inevitable.

Chaos

Let's say we get past resistance (which is by no means certain). Management buys in or issues the "Though shalt XP" edict. What happens next? Chaos — total, unrestrained, dizzying, performance-destroying chaos. We shouldn't be surprised, since that's exactly what the model predicts, but we always are. After all, we want XP to be a solution; we *need* to see XP as a solution, so we have a hard time when at first it appears only as a problem. And let's face it, at this point, we just aren't very good at it.

During chaos, the team wildly under- and overestimates. An iteration delivers zero functionality. Productivity and predictability quickly degrade under a lack of testing or refactoring. Under stress, the social fabric of pair programming disintegrates. The team finds their tools inadequate and spends time making better ones or replaces existing technology. Finger-pointing ensues about a miscommunication of a detailed requirement.

The chaos extends to the team's relation to the rest of the organization. When the team predicts a quarter of the way through a release that some scope will have to be dropped, they are labeled whiners or uncommitted. When they insist that the business folks hash out priorities before development continues, they are labeled troublemakers. When they ask the customer team to specify acceptance tests, they are accused of not caring about quality.

Individuals also experience chaos. Team leads, architects, and project managers whose value and contribution to the team have

been defined around the OSQ often have trouble finding their fit on an XP team (yes, their skills are still very valuable). Authority and responsibility are shifted to the team from "senior" individuals, causing them to experience a personal chaos and fear.

Something clicks, and things start to get better. Yes, it hurt. Yes, it was hard, but the worst is in the past.

As we move through chaos, the team and the organization are at great risk of reverting back to the old status quo. We often hear stories of teams that have abandoned XP after some initial bad iterations. Those that have not found their place and are still threatened by the change will escalate to upper management and threaten to leave the organization. Factions may form within the team to undermine the change agents. Politics can be rampant as people struggle to find a new comfort zone within the new set of practices.

Clearly, we will need a strategy to swim up this stream.

The Transforming Idea and Integration

If we successfully manage chaos, then, with some luck, a transforming idea will come along. Some examples we have seen:

- The team delivers software after months of schedule slips.
- The bug rate drops to levels thought to be unachievable.
- "I never have to use the debugger anymore!"
- The team embraces a single practice, often test-first development.
- An iteration goes particularly poorly, and the answer is obviously poor execution of the practices.
- A team member has a personal realization that "my old role really wasn't helping deliver the project."
- An individual confronts the "leave the team" decision and accepts that it is not his or her best option.
- Team members have their first good pair programming experience.

Something clicks, and things start to get better. Yes, it hurt. Yes, it was hard, but the worst is in the past. We integrate the practices and begin to see performance become more stable and predictable. Integration is the stage through which the odds of achieving the new status quo progressively get better than the odds of reverting back to the old status quo.

Practice

Project success requires a skilled team. Skills are developed and honed through practice. This stage is critical to ensure that the investment made in the change yields the highest return. The team begins to learn from and adapt the practices based on retrospectives performed after each iteration. Other organizations touched by the team learn to trust the style of communication and influence. Skills get better from the repetition and feedback. Eventually, teams settle into a fairly stable new status quo, where they are recognizably "doing XP" but with local adaptation and continual tweaking.

New Status Quo

We emerge from the transition at a new level of performance, typically characterized by lower defect rates, fewer change requests, and happy customers. The team is still susceptible to other foreign elements, some of which should be rejected, some of which should be embraced. XP is simply the current change focus for the team. There will be others, be they personnel changes, economic changes, competitive pressure, or one of many others. But as we shorten our project cycle times, the extent of the change required also shrinks. Over time, the team becomes increasingly comfortable working their way through the Satir Change Model.

NO EASY ANSWERS

Much as we'd like to trumpet XP as Good News, if we want to be successful, we'd better be prepared for the Bad News. Here are some things you can do in each of the stages:

Old status quo. Help the team become aware of the practices in XP and separate the hype from the reality. Create pull for the claimed benefits of XP in the team's customers. Avoid advocating why XP

is good for you and focus on why it is good for those whose support you need.

Resistance. Measure just how bad things are now. Get people to try experiments instead of intellectualizing about why it can't possibly work. The short (typically two-week) iterations of XP make process experiments easy to implement. Let the process tell you what works. Correct the common misconceptions about Extreme Programming [2].

Chaos. Proactively install organizational support for the change. Use change infrastructure models such as the Organizational Infrastructure for Change proposed by Shoji Shiba [4]. If problems have been caused by not sticking to the rules, try going back to the rules. Support an XP user's group so the team can see that their problems are more universal. If the team is stagnating, get help. Communicate. Communicate. Communicate.

Because moving to XP requires changes to the social relationships of work, it can't possibly be easy — and it isn't.

Integration. Incubate transforming ideas. These will often be forms of feedback, so overdo feedback as the team begins the transition. Use the measurements developed during resistance to demonstrate improvement. Hold a half-hour retrospective every iteration to propose and evaluate process experiments. Hold two-day professionally facilitated retrospectives every quarter.

New status quo. Encourage team members to move between teams to continue cross-fertilizing ideas.

When Lowell first gave a talk based on this material, one of the comments was, "Lowell didn't say a single positive thing all night." So be it. Because moving to XP requires changes to the social relationships of work, it can't possibly be easy — and it isn't. Successful XP transitions involve heavy communication and being proactive about the changes that will be experienced at all of the levels.

What has your experience been in transitioning to Extreme Programming? Contact the authors with your story.

REFERENCES

1. Beck, Kent. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000.

2. Jeffries, Ron. "The Business of Extreme Programming." Cutter Consortium Agile Project Management *Executive Report*, Vol. 3, No. 2, 2002.

3. Satir, Virginia et al. *The Satir Model*. Science & Behavior Books, 1991.

4. Shiba, Shoji et al. *Four Practical Revolutions in Management*. Productivity Press, 2001.

5. Weinberg, Gerald. *Quality Software Management*: *Anticipating Change*. Volume 4. Dorset House, 1997.

Lowell Lindstrom is Founder and President of The Oobeya Group, LLC., a services firm that helps technology business achieve success in rapidly changing markets. He was previously Executive V.P. at Object Mentor, Inc., where he is responsible for business affairs and coaching organizations on customer skills and organizational change. Prior to Object Mentor, Mr. Lindstrom worked for Teradyne, Inc. where he was involved in all aspects of successful commercialization of software systems. After many years of developing and managing software projects, Mr. Lindstrom worked in various general management, marketing, and sales roles with Teradyne's software test businesses. He has a bachelor's degree in computer science from Northwestern University and a master's degree in management from Northwestern's Kellogg School of Management.

Mr. Lindstrom can be reached at lowell@oobeyagroup.com

Kent Beck is a Senior Consultant with Cutter Consortium's Agile Project Management practice; a regular contributor to Cutter IT Journal*; and a keynoter, panelist, and workshop leader at the annual Cutter* Summit*. Mr. Beck is the founder of Extreme Programming and author of numerous books, including* Extreme Programming Explained: Embrace Change, Smalltalk Best Practice Patterns*, and, with coauthor Martin Fowler,* Planning Extreme Programming*. Mr. Beck has been a programmer, primarily in Smalltalk, for over 17 years. He has pioneered CRC cards, the HotDraw drawing editor framework, the xUnit family of testing frameworks, patterns for software development, and the rediscovery of test-first programming.*

Mr. Beck can be reached at kbeck@cutter.com.