CUTTER IT JOURNAL

The Journal of Information Technology Management

CUTTER CONSORTIUM
ACCESS TO THE EXPERTS

September 2003 Vol. 16, No. 9

Patterns in Software Development

An Idea Whose Time Has Come
Software patterns have been helping us design better software for many years. They capture and effectively transmit highly useful knowledge that was once solely in the minds of gurus.

Ah, Forget It
Most people who think they're using patterns really aren't. Tool support for patterns has been a failure, and the first truly great pattern language has yet to be written. If this is as far as we can get in 10 years, it's time to pack it in.

opinion

"Alexander reminded the crowd that the real purpose of any pattern language must be nothing short of improving human existence. That was a lot for a stadium-sized crowd of engineers to swallow."
— Joshua Kerievsky, Guest Editor

Singing the Songs of Project Experience: Patterns and Retrospectives

by Linda Rising and Esther Derby

"You must never feel badly about making mistakes," explained Reason, "as long as you take the trouble to learn from them. For you often learn more by being wrong for the right reasons than you do by being right for the wrong reasons."

— Norton Juster, *The Phantom Tollbooth* [6]

Have you ever been on a software project that you'd want to repeat in exactly the same way? Neither have we.

But on every project, some things that go well — some actions and practices — are worth repeating. Often the impacts of those critical choices and actions pass unnoticed, except by the person who chose a particular alternative or course of action. Those insights aren't part of the group conversation. As a result, the individual may learn, but the benefit of the learning is confined to that individual's sphere of influence.

How can organizations expand the benefits of hard-won experience? Between us, we have well over a decade of experience working with teams and organizations to capture critical learning from project experience. We'll share what we've learned about capitalizing on group learning and disseminating knowledge through the powerful combination of project retrospectives and patterns.

A SIMPLE IDEA: LEARN FROM THE PAST

Here's how our friend Norm Kerth, quoting *Winnie-the-Pooh* author A.A. Milne, describes the state of many people working on software projects [7]:

Here is Edward Bear, coming downstairs now, bump, bump, bump, bump, on the back of his head, behind Christopher Robin. It is, as far as he knows, the only way of coming downstairs, but sometimes he feels that there is another way, if only he could stop bumping for a moment and think of it.

If we just took time to reflect on *our* experiences, maybe, just maybe, we could figure out a better way!

In 1988, Joseph Juran identified retrospective analysis as a method of learning from work experiences. At that time, he called the practice a "Santayana review" in homage to the great philosopher George Santayana [5]. Since that time, many organizations have taken up the practice in many forms and under many different names.

The idea is simple — take time to examine what happened on the last project and learn from it. Teams using agile methods are beginning to incorporate retrospectives at the end of each iteration. As Dave Parnas has said, "We want to learn about completing projects while we are completing projects."

When we work with teams in project retrospectives, we help the group to see the "big picture" of the project, understand how their work fits in with the whole, and gain insights about the work they did together. Team members share specific techniques and practices that helped them to succeed. We guide people to work with the insights and data they've generated to identify gaps, analyze root causes, and suggest changes and improvements for the next project. A retrospective is an opportunity to learn from what worked and what didn't work.

When we work with teams in the months after a project retrospective, the folks we talk to are specific about what they personally are doing differently and the changes the team has implemented. Teams that have participated in retrospectives mention better communication and greater awareness of how their work affects others.

Obviously, the importance of knowledgeable people and the sharing of knowledge are not new. Better individual and corporate knowledge has always paid off. What's new is the current level of attention to this topic. We know that knowledge is no longer the business of an elite few but has become the "business of every business." Today knowledge is power, and organizations are struggling with the best ways to manage knowledge, their most precious resource.

Over time, people develop contextual knowledge — the "know-when" about using the "know-how."

We know that just accumulating individual, or even team, knowledge is not enough. The advantage resides in sharing knowledge among other members of the workplace, across teams. We're interested in what people know and how to keep them knowledgeable.

PATTERNS CAPTURE PROVEN SOLUTIONS

People don't make bad decisions or apply "solutions" that don't work because they want to fail or because they are stupid. In most situations, people consider the available alternatives and choose the one they believe will be most successful.

The trouble is that often people have a limited repertoire of alternatives. And when they have a limited set of alternatives, they are more likely to choose an alternative that worked before, even when the current circumstances are quite different.

One remedy for this is, of course, experience. Over time, people make adjustments, try new alternatives, and develop a sense for when to choose alternative A over alternative B. Over time, people not only expand their repertoire of alternatives, they develop contextual knowledge — the "know-when" about using the "know-how."

But life is too short to learn everything we need to know from experience — we need to learn how to benefit from others' experiences, too. The patterns community is trying to capture expertise that resides in experts' heads, document it, and share it. A pattern is simply a way of documenting a solution to a recurring problem so that the solution is accessible across teams and even across organizations.

RETROSPECTIVES + PATTERNS = ACCELERATED EXPERIENCE

Retrospectives (or postmortems, project reviews, or whatever you choose to call them) are a powerful way to increase individual and organizational capability. In addition to helping teams develop proposals for doing things differently next time and suggesting improvements, project retrospectives provide a wonderful opportunity for capturing knowledge as patterns. The team is there and everyone is focused on learning — what better time to identify patterns?

Up until now, individuals have captured best practices as patterns by mining their own experience. That is, a pattern author with expertise in a particular domain will recall a particular problem that he or she has successfully solved over and over. The pattern author will then attempt to capture information about that problem-solving experience.

During a retrospective, the project team can identify likely patterns, and small groups may write a pattern outline as part of processing what they've learned. Groups that are doing similar functional work are the most likely to see patterns in their work. After forming affinity groups of three to five people,[1] ask them to think carefully about the project and identify two or three of the project's most critical moments. A critical moment can be a decision, a turning point, or an action that overcame an obstacle or made a difference in some other way.

In 15-20 minutes, most groups can sketch out the key elements of a "proto-pattern." Groups then

[1]If there are more than five people, break them into two or three groups.

present their proto-pattern to the rest of the team and answer questions to clarify forces, solutions, and resulting context. When teams share solutions during retrospectives, they learn that they find useful information by looking at what other teams and other people are doing.

Sometimes two or more teams identify what turns out to be the same pattern. These might be candidates for formal patterns.

Within a pattern, the forces, resulting context, rationale, and known uses sections document the relevant information. The pattern not only explains the problem and the solution but also provides guidance on when the practice will be effective and what the result of using the pattern is likely to be. Pattern writing within a retrospective encourages teams to consider the "know-when." To see how this works, let's consider a pattern mined from an actual project retrospective.

NO MORE THAN 10

No More Than 10 is an example of a pattern written in one organization. Remember, patterns are the best practices of other organizations, so pay attention!

Pattern Name: No More Than 10

Problem: How many people should be on a team to ensure effectiveness?

Context: Product development with limited resources and a tight schedule.

Forces

Too few people and you won't meet the schedule.

When there are too many people, communication overhead increases.

There may be pressure from customers or management to add more people.

There's always a penalty when another person is added to the team. If enough people are added, progress can stop completely.

Solution

No more than 10 people should be on a team. If the project is large, you may have several teams. Interfaces between the teams should be clearly defined, and each team should be responsible for a cohesive part of the work.

Resulting Context

The small teams can function relatively independently and make the best possible progress toward the delivery date.

Having a minimally sized group can be risky if the team is in danger of losing any member.

Rationale

According to Cutter Consortium Senior Consultant Alistair Cockburn, "Separating workers into smaller clusters calls for your architects to partition the system so that teams of three to five people can [behave] exactly as on a small project" [3].

Fred Brooks states, "The ideal ... [is] the small, sharp team, which by common consensus shouldn't exceed 10 people" [2].

Pattern writing within a retrospective encourages teams to consider the "know-when."

Brooks also includes the following explanation of Conway's Law (documented as a pattern by Jim Coplien):

> Organizations that design systems produce systems that are copies of the communication structures of the organizations. ...
>
> The organizational structure and the structure of the system are mirror images. Decisions driving the creation of one can affect the other. A clean, well-designed system with minimal interfaces between the subsystems reflects an organization where each small team works on a subsystem or a well-defined piece of a subsystem, where each team and each piece of the system are relatively independent [4].

Known Uses

The following are statements from project development retrospectives:

> I try to estimate the amount of work to keep this team of 10 busy. I think the maximum size is 10. More than 10 is too much overhead.
>
> The project had about 10 people on it, average including System Test and Marketing.
>
> The team had six to eight developers and two system testers, a

small team that interacted daily. They sat close together and worked together.

The team was originally six to seven people. Now there are 30-35. When you have a small group, it's OK to be self-directed. With a larger group, there are too many given the amount of change. It can't be managed. It requires a team of managers.

Team chemistry was very good. There were five to 10 people who worked well together, even those with diverse/clashing personalities.

We kept the team size small, around nine, always less than 10. There might have been pressure from management to add more people to try to get it done faster, but we didn't want to add any more people. The tenth person would have made it difficult to divide the work.

Related Patterns

This rewriting of Jim Coplien's Size the Organization pattern [4] has a new name and the project retrospective data to make the pattern more compelling.

ROLL-YOUR-OWN PATTERNS

When retrospective data from successful teams shows that a team size of no more than 10 is a factor in the success of the project, and when observations by Alistair Cockburn, Fred Brooks, and James Coplien back up the pattern, you know it's an important pattern. Capturing this important information and naming the pattern No More Than 10 is a useful way to ensure that this knowledge is not lost.

You might be saying that this seems obvious. Of course it is! At the end of every project, when your belief in good software practices prompts you to hold a retrospective, what are the lessons learned? Are they startling revelations? Probably not. They're probably the same non-startling revelations you had on the last project, and the project before that, and so on. But if we don't pay attention and take time to reflect, we don't bring this common sense into our ordinary conversations.

The pattern name will become part of the organization's vocabulary, so choose wisely.

Here's a basic pattern form with which you can begin to formulate your own patterns:

Name: Think of a word or short phrase that captures the intent of the pattern. This name will become part of the organization's vocabulary, so choose wisely. Sometimes the users will already call the solution by name.

Context: Describe the setting in which the problem occurred, including any other patterns that had been applied.

Problem: Specify exactly what the problem was.

Forces: Explain why this problem was so hard to solve. Sometimes novices don't understand why a particular problem is really difficult.

Solution: Give enough detail so that a novice can solve the problem, but describe the solution at a high enough level that you "can use the solution a million times over, without ever doing it the same way twice" [1].

Rationale: Convince the reader that you know what you're talking about. You might bring in outside references to back up your experience. Tell a story!

Resulting context: Every solution has side effects and creates new problems when it is applied. Prepare the reader for this.

Known uses: A pattern should have been applied in at least three different settings. Cite these with a little explanation of how it worked.

Related patterns: Usually a pattern has many connections to existing patterns. Reference these patterns and explain the connections.

Author: Cite the author. Usually the author is just the person who writes down the information; the knowledge itself may have come from many sources. We usually give credit to the author for the documentation, but the author is free to acknowledge mentors or teachers.

There are many pattern forms. While this "canonical" version probably won't produce a literary masterpiece, it will capture the essentials. If the solution is worthwhile, the pattern will be useful and used. To produce a really well-written pattern, an author should have good writing skills and an interest in taking the pattern through writers' workshops, where he or she can get feedback on the pattern in order to improve the message. Many feel that, at some point, a transition to Alexandrian form is appropriate, but this is up to the individual writer. The point is, pattern writing is challenging and takes time. To produce something that will effect change outside the organization, more than just an outline is required. Inside the organization or the team that identified the pattern, however, capturing the essence is probably good enough and requires little investment.

WHEN TO WRITE PATTERNS (AND WHEN NOT TO)

When a process owner is creating patterns, it's typically because he or she has seen some successes across several projects and is concerned that this knowledge is not widespread. If the success is based on something everyone knows and already incorporates, there's really no reason to spend time capturing that. Writing patterns is hard work, so you want to be sure that you will get a good return on your investment. As one observer once said:

> If there isn't a horror story or a pot of gold story behind the idea, why burden the organization with it? There is no need to formalize common sense when it works just fine [9].

You also don't want to spend time writing a pattern if you have no known uses of it. A pattern is not just a theoretical proposition; it must be grounded in experience. A pattern is something that has worked over and over and over again, not just a good idea for solving a dilemma.

So you want a good idea with several known uses — a documented insight that will prevent others from failing to apply the solution. This must be the job of the process owner. He or she sees the result of the project retrospectives or attends them all. This individual must also have the chance to attend project planning meetings or other meetings to make sure the word gets out. Having the knowledge and spreading the word are two different tasks.

As we have discovered, the power of patterns is that they provide a resting place for the stories captured in retrospectives and contributed by others (for example, in a writers' workshop). We believe that a pattern can help even if the idea never spreads beyond the team that identified it and even if the pattern remains a fledgling. The pattern No More Than 10 has "saved" a lot of projects even though it's not a polished piece of writing.[2]

[2]For an example of a pattern that's been polished and translated to the Alexandrian form, see "Just Say Thanks" in M.L. Manns and L. Rising, *Fear Less and Other Patterns for Introducing New Ideas into Organizations* (Addison-Wesley, forthcoming 2004).

Someone needs to sing the songs and tell the stories of successful solutions to devilishly recurring problems — a patterns minstrel, if you will.

SHARE THE WEALTH OF EXPERIENCE

Patterns and insights won't permeate the organization on their own. Someone needs to sing the songs and tell the stories of successful solutions to devilishly recurring problems — a patterns minstrel, if you will.

Who should be the patterns minstrel? It depends. In a company organized around functions or centers of excellence, the local process owner may own the patterns relevant to that function or expertise (e.g., testing, design, project management). In organizations with a quality assurance function, one of the QA members may take the responsibility for patterns. In agile teams, it may be the coach or Scrum Master. The point is that there must be some person who documents and educates and reminds people of relevant patterns — not your mother reminding you that a stitch in time saves nine, exactly, but someone who can be the local deputy of the chief knowledge officer (even if you don't have one).

It does take time and resources to disseminate knowledge across teams. Nothing worthwhile is free.

The patterns don't write themselves, and patterns can't sing their own song.

Here are some ways for a resident minstrel to sing the songs across the organization.

Post patterns on an intranet, and as related patterns are collected and organized, they can be presented in patterns training to the appropriate audience. When patterns include the author's name and list contacts in the known uses section, people have a place to go for more information.

Host a monthly tech forum to periodically update the organization on the latest patterns captured in project retrospectives. In one company we know, pattern owners present new patterns and share the stories behind the patterns. This company is helping people not only to learn from their own experience, but also to benefit from the experience of others.

Speak the language. As people across the organization learn the patterns, they will speak the language appropriate for their domain: design, test, customer interaction, project management. The names of the patterns become part of the local domain vocabulary. The names reflect the known uses and stories in the patterns and are recalled whenever anyone uses a pattern name.

Help the newbies learn the lingo. There's no need to explain why a pattern is appropriate when everyone speaks the same language. In one company, the newbies are always flushed out by the language, as is any newcomer in a corporate setting who doesn't know the local buzzwords. The new person needs someone (or someplace) he or she can go to for help in understanding what the team is talking about. This help can come from the pattern owner, a training class, a Web site, copies of some papers, or just a little one-on-one mentoring.

The XP coach or Scrum Master (even a PMI-certified project manager!) can be the patterns minstrel for the teams he or she works with. An informal review of relevant patterns can become part of project planning and even risk management. Start by asking these questions:

- What challenges are we likely to face?
- What solutions have worked in the past?
- What patterns do we have that might help us address this potential problem?

The pattern owner could be a local guy who just likes patterns. Many organizations start out with one or two evangelists [8] holding brown-bag sessions to educate coworkers about patterns and share new patterns. The downside of such an informal approach is that if there's no recognition of the importance of the role — it takes too much work for a spare-time activity — it will die.

THE SAVING POWER OF PATTERNS

Patterns document the lessons learned, capture the context, and provide a name for useful solutions. Then, in the heat of battle, when we're pushed to fall back on some tried and not-so-true practice like adding more people, someone can say, "Hey! No More Than 10!" and we can remember stories — ghosts from the past. Those stories speak to us and can save us. We can recall all the good things we know, and maybe this time we'll get it right — or at least a little more right!

Retrospectives are a natural place to look for patterns. Group pattern generation focuses team members on increasing their repertoire of context-appropriate alternatives. For companies that are concerned about becoming learning organizations, this is a step in that direction.

REFERENCES

1. Alexander, Christopher, Sara Ishikawa, and Murray Silverstein, with Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language.* Oxford University Press, 1977.

2. Brooks, F.P. *The Mythical Man-Month*, 2nd ed. Addison-Wesley, 1995.

3. Cockburn, A. *Surviving Object-Oriented Projects: A Manager's Guide*. Addison-Wesley, 1998.

4. Coplien, J.O. "A Generative Development-Process Pattern Language." In *Pattern Languages of Program Design*, edited by J.O. Coplien and D.C. Schmidt. Addison-Wesley, 1995.

5. Godfrey, A. "The Santayana Review." *Quality Digest*, February 1999.

6. Juster, N. *The Phantom Tollbooth*. Knopf, 1961.

7. Kerth, N. *Project Retrospectives: A Handbook for Team Reviews.* Dorset House, 2001.

8. Manns, M.L., and L. Rising. *Fear Less and Other Patterns for Introducing New Ideas into Organizations*. Addison-Wesley, forthcoming 2004.

9. Moneymaker, P., and L. Carter. "Managing the Invisible Aspects of High-Performance Teams." *Crosstalk*, May 2001, pp. 29-33.

ADDITIONAL READING

Davenport, T.H., and L. Prusak. *Working Knowledge: How Organizations Manage What They Know.* Harvard Business School Press, 1998.

DeLano, D., and L. Rising. "Patterns for System Testing." In *Pattern Languages of Program Design 3*, edited by R. Martin, D. Riehle, and F. Buschmann. Addison-Wesley, 1998.

DeMarco, T. *The Deadline*. Dorset House, 1997.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

Taylor, P. "Capable, Productive, and Satisfied: Some Organizational Patterns for Protecting Productive People." In *Pattern Languages of Program Design 4*, edited by N. Harrison, B. Foote, and H. Rohnert. Addison-Wesley, 2000.

Linda Rising has a Ph.D. from Arizona State University in the area of object-based design metrics. Dr. Rising's background includes university teaching experience as well as work in industry in the areas of telecommunications, avionics, and strategic weapons systems. She has been working with object technologies since 1983. Dr. Rising is the editor of A Patterns Handbook, A Pattern Almanac 2000, *and* Design Patterns in Communications Software. *She has published numerous articles in industry publications and is a regular contributor to* DDC-I Online News *(ddci.com/news_latest_news_archive.shtml).*

Dr. Rising has presented a number of tutorials and workshops at JAOO, OOPSLA, *and other conferences. She is currently coauthoring a book with Mary Lynn Manns:* Fear Less and Other Patterns for Introducing New Ideas into Organizations *(Addison-Wesley, forthcoming 2004).*

Dr. Rising can be reached at E-mail: risingl@acm.org; Web site: www.lindarising.org.

Esther Derby is founder and president of Esther Derby Associates, Inc. a management consulting firm based in Minneapolis, Minnesota. Ms. Derby works with software project teams to start projects on a solid footing, assess the current state of the project, capture lessons learned, and help the team apply these lessons to the next iteration or project. She also coaches and trains technical people who are making the transition to management.

Ms. Derby is technical editor of STQE *magazine and a regular columnist for Stickyminds.com and Computerworld.com. She publishes the acclaimed quarterly newsletter* insights. *Ms. Derby has a B.A. from the University of Minnesota and an M.A. in organizational leadership from the College of St. Catherine.*

Ms. Derby can be reached at Esther Derby Associates, Inc., 3620 11th Avenue, Minneapolis, MN 55407, USA. Tel: +1 612 724 8114; E-mail: derby@estherderby.com; Web site: www.estherderby.com.

Cutter IT Journal

Topic Index

Joshua Kerievsky, Guest Editor

Joshua Kerievsky, a Senior Consultant with Cutter Consortium's Agile Project Management Practice, began his career as a professional programmer at a Wall Street bank, where he programmed numerous financial systems for credit, market, and global risk departments. After a decade at the bank, he founded Industrial Logic to help companies practice successful software development. Mr. Kerievsky has programmed and coached on small, large, and distributed XP projects since XP's emergence. He recently pioneered Industrial XP, an application of XP tailored for large organizations. Mr. Kerievsky has been an active member of the patterns community and is presently completing a book entitled *Refactoring to Patterns*. He can be reached at jkerievsky@cutter.com.

CUTTER IT JOURNAL

Upcoming

Issue Themes

Usability

Metrics and Benchmarking II

Killing IT Projects

http://www.cutter.com/ or +1 800 964 5118