Patterns Mining

by
Linda Rising
AG Communication Systems

Introduction

The Design Patterns text [Gamma, 1994] appeared in October 1994 and started a wave that swept across the software community. At first there were patterns for object-oriented software and now there are analysis patterns, system test patterns, organization and process patterns, Smalltalk patterns, customer interaction patterns, patterns for many domains and interests.

A pattern is a way of documenting experience by capturing successful solutions to recurring problems. This documentation is complete in the sense that the context for the problem, the forces that weigh upon the problem-solver, the rationale and the resulting context of applying the solution, must all be carefully recorded. A pattern is more than a simple heuristic that leaves users without a sense of when it is appropriate to follow the guideline. This additional information makes the pattern useful. An example of a pattern template can be found on the AG Communication Systems Patterns Home Page [AGCS]. The template shows the information that must be documented to produce a useful pattern.

What every corporate group in the world wants, however, is not a book of patterns written by someone outside the company, but a collection of patterns that reflect the company's way of doing business. The business goal is a handbook of best practices that documents the things the company has done well. This handbook would ensure that corporate wisdom would be recorded and shared with current and future employees. It is almost certain that no one person already has all this knowledge, so everyone in the company would become more knowledgeable about the business. The whole organization would improve.

This sounds wonderful, but how is it accomplished? Obviously, such a handbook does not appear overnight but requires considerable investment. Will the people who know the business willingly create patterns and, in so doing, perhaps, make themselves less valuable to the organization? In these days of downsizing and reengineering, this may not be likely. Will roving patterns reporters identify the corporate citizens with the requisite wisdom and coerce them into revealing their secrets? How will these pattern gatherers identify knowledge holders, interview them, write useful patterns, document them in a handbook, and ensure that the knowledge is successfully transferred to others?

These are open questions. People in and outside the patterns community are interested in answers, but, in the meantime, some of us are experimenting in our own companies. We are patterns pioneers, striking out to learn from our own experience.

In March 1995, I presented the notion of patterns to our vice president and his staff. There was almost unanimous approval for the idea and a real interest in a corporate handbook of best practices patterns. Since that time, along with David DeLano [DeLano, 1997], my fellow patterns miner, we have tried to solve the problem of developing this handbook in many different ways. This article will share some of our successes and failures and hopes for the future of our work.

First, let me introduce my company. AG Communication Systems is based in Phoenix, Arizona. It is a joint venture of Lucent Technologies and GTE and is a leading developer and manufacturer of advanced telecommunications products and services, including switching, intelligent network, access and wireless products. The company's core product, the GTD-5 digital central-office switching system, has an installed base of more than 17 million lines serving business, government and industry, as well as residential subscribers serviced by the public telephone network. AG Communication Systems employs more than 2,000 people, about 700 of whom are directly involved in software development.

My role at this company is an unusual one. I'm the patterns champion or the self-proclaimed patterns "princess." I "do" patterns all day long! In the following chapter I will share my experience by describing the following techniques: Mining by Interviewing, Mining by Borrowing, Mining by Teaching Patterns Writing, Mining in Workshops, Mining Your Own Experience, Mining in Meetings, and Mining in Classes.

Underlying Principles

Patterns as a technology is in its infancy with few theoretical underpinnings. Most work published about patterns is based on practice. Indeed, the whole thrust of patterns derives from experience. The Design Patterns text [Gamma, 1994] is a collection of object-oriented design best practices that shares what four expert designers have seen. There is no presentation of design methodology or theory. It is a catalog of twenty-three patterns that have solved real problems in many settings.

Those of a theoretical bent have been looking to undergird the patterns movement with a formal infrastructure, just as formal methods investigators have been attempting for decades to similarly make software engineering tractable. Both movements have been unsuccessful in making inroads in the software development community. At this time, no theoretical approaches have been developed for patterns mining. It is an attempt by practitioners to document their best practices.

Best Practices

Mining by Interviewing

The first pattern mining efforts at AG Communication Systems were begun after a visit from Jim Coplien and Bob Hanmer of Bell Labs. They had been doing pattern mining with the developers of the 4ESS®, a member of the AT&T electronic switching system product line. This activity was in response to a concern by AT&T management about the imminent retirement of some long-time developers on the 4ESS. Their pattern mining took the form of interviewing these experts and recording the information they gathered in pattern form. Over 100 patterns have been mined. Some have been documented in the Addison-Wesley PLoPD series [Adams, 1996].

We were inspired to experiment with this mining tactic. We taped one-hour interviews with a couple of experts on the GTD-5®. We returned to our desks to translate what was recorded and document that information as a pattern. After the patterns were documented, we tried, with varying degrees of success, to return to our sources to verify that what we had written really captured the speaker's intent. This is a long, slow, difficult process.

Usually the company gurus, the people everyone recognizes as having the best domain understanding and the greatest knowledge of company history, are extremely busy people. Most developers are tied to a project, and when that project finished, they have a little breather between projects and a little time as the next project ramps up. Gurus are always tied to several projects. They are in constant demand. They never have a breather. Even though they recognize the need to transfer what they know to others, they often do not have the time for the interview and the subsequent review. Ralph Johnson once suggested [1996] that we take these people off-site, away from interruptions, for a day or a week-end. This suggestion has been documented as one of a collection of patterns for introducing patterns into the workplace [OOPSLA, 1996]. This solution is difficult to implement. Gurus have no free time, even for trips at company expense!

The related problem with interviewing gurus is asking the right questions. Some we have used: What would you share as a mentor? What would be lost to the company if you left tomorrow? What problems have you solved successfully on several projects? What do you say repeatedly at project meetings that never gets documented?

Gurus are strong, extremely knowledgeable people and steering them toward useful information takes skillful interviewing techniques. I don't feel that we have learned how to do this effectively. On the few occasions when we have shared an hour with a guru, the information we have gathered has not been as useful as we had hoped.

A positive result of our experience is that we have learned that everyone is anxious to share their knowledge. No one seems to be afraid of becoming less valuable to the company. In fact, sharing successes makes these contributions visible, when, in many cases, that accomplishment would have been lost completely. These stories are a valuable part of the patterns. Roger Shank has written a fascinating book that helps explain this [1990]. He states:

"People need to talk, to tell about what has happened to them, and they need to hear about what has happened to others, especially when the others are people they care about or who have had experiences relevant to the hearer's own life."

Stories are documented in the pattern rationale. Stories always draw readers in. They love to hear a personal account of what went wrong or what went right on an earlier release of a product. The people at my company have a tremendous respect for each other. I think that's why the patterns work has been so successful. Shank supports our observation:

"We can tell people abstract rules of thumb which we have derived from prior experiences, but it is very difficult for other people to learn from these. We have difficulty remembering such abstractions, but we can more easily remember a good story. Stories give life to past experience. Stories make the events in memory memorable to others and to ourselves. This is one reason why people like to tell stories."

Using patterns to mine patterns is very effective. When you tell people about a pattern, invariably, they will have a story about the pattern, which can be added to the pattern document, or they will have a disagreement with the solution. Usually this means the context is too broad. The story the dissenter will tell helps narrow the context and improve the pattern. Context in a pattern is related to Shank's statement about contexts for stories:

"People need a context to help them relate what they have heard to what they already know. We understand events in terms of events we have already understood. When a decision-making heuristic, or rule of thumb, is presented to us without a context, we cannot decide the validity of the rule we have heard, nor do we know where to store this rule in our memories. Thus, what we are presented is both difficult to evaluate and difficult to remember, making it virtually useless. People who fail to couch what they have to say in memorable stories will have their rules fall on deaf ears despite their best intentions and despite the best intentions of their listeners. A good teacher is not one who explains things correctly but one who couches explanations in a memorable format."

Mining by Borrowing
Since AG Communication Systems creates telecommunications systems and since AT&T (now Lucent Technologies) was our parent, we were privileged to share in the patterns that Coplien and Hanmer had gathered on the 4ESS. We are examining these patterns and learning what we can from them. Mining by borrowing is appropriate for companies in the same domain who can share information. Unfortunately, except in a joint venture setting, this is a rare occurrence. Domain knowledge is regarded as proprietary, even when the information is of a truly general nature. Since domain-specific patterns are being published, these patterns provide a base that can be extended with proprietary patterns.

I recently attended a seminar by Peter Senge, author of The Fifth Discipline [1994]. We talked about the problem of companies' sharing information and he told me an interesting story. Senge had been working as a consultant with Ford. There had been some successes as a result of this work. A high-level executive at Ford had suggested sharing these successes with Chrysler. His fellow executives were horrified and asked him why he would even consider sharing information about success with a competitor. The answer is especially important for members of the patterns community. He said he couldn't image not sharing, because that would hamper their own success. Senge's new approach required not only a change in the way Ford operated but also a change in attitude toward Chrysler. To translate this to patterns, sharing what we know with others increases our own understanding. In many cases, we find that what we have learned is not domain-specific but applies across the industry. We share what we know and we all improve. Drawing a boundary at organizational lines hampers the work of the whole community.

This is a new way of doing business! Clearly, we are not giving away trade secrets. Details of product information are not handed over to competitors. Senge's notion of learning organizations and the changes it makes in the way we work and the ideas captured in patterns are solutions to problems that appear anywhere. To share the best of us with the rest of us means that we all improve.

David Armstrong's Managing by Storying Around [1992], is a collection of management observations that apply across many domains. The author is vice president of Armstrong International, a manufacturer with offices throughout the world. He has taken one of the oldest forms of communication, storytelling, and turned it into a powerful management tool. Here is his answer to the sharing question:

"If a story helps a competitor, so be it. A better competitor forces us to be better. Besides, what I really think will happen is our competitors will read our stories and become frustrated because they can't do business the way we do."

Mining by reading is a form of mining by borrowing. Books by experienced people are fertile ground for patterns mining. In April 1996 we enjoyed a visit from Jim McCarthy, author of The Dynamics of Software Development [1996]. Jim was not familiar with patterns but his book contains a collection of guidelines that resemble patterns. To illustrate the close relationship, I translated one of his guidelines to a pattern. The result, "Don't Flip the Bozo Bit!" can be found on our external web site [AGCS]. It describes the problem of software development teams casting someone as a bozo. After that, no one listens to that person, so the team has lost a contributor. The guideline warns that everyone must be a part of the development effort and that team members must be constantly aware of this possibility and "get everyone's head into the game!" McCarthy also cautions us about flipping the bozo bit on ourselves. Other patterns I have documented are: "It's a relationship, not a sale!" and "Enrapture the customer!" These are included in a collection of customer interaction patterns on our external web site [AGCS].

Mining by Teaching Patterns Writing
Coplien and Hanmer brought not only the report of patterns mining on the 4ESS but also a patterns writing class they had been teaching at sites within AT&T. By learning how to write patterns, participants learn a lot about patterns and they also produce one. This is an effective way to start patterns mining in an organization. Some writers are so eager, especially if appropriate reward mechanisms are in place, that they continue to write after the class has ended. At AG Communication Systems, we reward pattern writers with a patterns publication. After several patterns have been written, we look for publishing opportunities for the enthusiastic contributor. This approach is appealing to frustrated writers within the development community. Unfortunately, these folks are a minority. Most engineers do not like to write. As Ralph Johnson has noted [1995]:

"You'll find that most people are not good at finding patterns. Fortunately, you don't need (or want) most people to be doing that. You want to make everybody feel included and wanted, but ultimately people will sort themselves out into those who will want to work on finding patterns and those who don't…your organization's real job is to get a product finished, and patterns are a means to that end. Most people should just get on with their work."

Sometimes pattern writers come from unlikely places. Most of those who have followed the patterns activity know that a building architect, Christopher Alexander, is responsible for the patterns movement. In his work, The Timeless Way of Building [1979], he writes:

"Every person has a pattern language in his mind. …Your pattern language is the sum total of your knowledge of how to build. The pattern language in your mind is slightly different from the language in the next person's mind; no two are exactly alike; yet many patterns, and fragments of pattern languages, are also shared."

Alexander was talking about building cities, towns, neighborhoods, and houses, but his words can apply to any product. Creation involves patterns that are, to some extent, shared by other creators. This phenomenon of shared patterns can be seen in a writers class where a pattern written by one person is recognized by another participant. There is a sense that the pattern captures an important or essential element of the domain. This feeling is a response to something Alexander calls the "quality without a name." This quality is what makes the pattern live. As Alexander notes:

"The fact is that we feel good in the presence of a pattern which resolves its forces. …people who come from the same culture do to a remarkable extent agree about the way that different patterns make them feel."

The patterns writing class introduces the writers workshop. This process is documented on our external web page [AGCS]. The approach was introduced to the patterns community by Richard Gabriel. Writers workshops, interestingly enough, are for writers. This helpful medium enables a writer to hear feedback from colleagues in a non-threatening setting. Everyone in a writers workshop should also be a writer and, therefore, equally sensitive to comments. Writing patterns is difficult and those who have struggled to capture their experience in a pattern are in a good position to help others who have chosen the same path.

Mining in Workshops

When Jim McCarthy visited us, he told the story of a problem-solving workshop technique used by management at Microsoft. Managers would gather around a conference table and someone would begin by describing a problem and asking for help in its solution. Someone else would propose a solution: "We had that problem on Project W and we resolved it by doing X." Another would respond, "Yes, we also had that problem and we did Y." After a few proposals, someone would summarize, "What we're talking about is Z!" At that, notebooks would open and managers would start writing. What was being observed and captured was the essence, a high-level extraction, of the solution to the problem. They're capturing patterns.

Always on the look-out for new pattern mining approaches, I thought we might try patterns mining in a workshop format. By happy coincidence, as we were debating this approach, we were also teaching a patterns writing course. A pattern written by Mike Sapcoe, a system tester, provided the opportunity we were seeking. I was glad to see Mike's system test pattern, our first in this area. Ray Fu, the coach for system test, was interested and supported us.

We sat in a room with a dozen of our best system testers and asked them to talk about recurring problems with known solutions. We typed furiously while the discussion raged around us. We were lucky to have the assistance of Greg Stymfal, a member of our patterns community with system test experience and effective facilitation skills. This role is critical in corralling a room full of old hands! The result of our trial with workshops is a paper that will be published in the pattern series by Addison-Wesley [Delano and Rising, 1997] and is also available on our web site [AGCS].

Since the time of publication, the patterns have grown considerably and continue to grow. Patterns are living things that change as we learn more about the problem, the context, the solution. Some may completely outlive their usefulness as procedures and other technologies change. These patterns should be gracefully retired as new ones spring up to replace them. As Alexander has observed [1979]:

"As people exchange ideas about the environment, and exchange patterns, the overall inventory of patterns in the pattern pool keeps changing. …Of course, this evolution will never end."

We grow the system test patterns by repeating the workshop experience, each time with a new set of testers. We present the existing patterns and ask for feedback, again, typing energetically to keep up with the comments. Everyone agrees with the solutions in the patterns and many testers can help us improve the patterns with new stories. We have also split some patterns into two or more smaller patterns and spawned new patterns. During this workshop process, the testers learn the existing patterns and we learn how to improve them. In this new training approach the trainers and the trainees both learn.

Our goal is to have all system testers go through the system test patterns workshop experience. We will then teach the patterns to our coaches and the rest of the corporate development community. This should ensure a more uniform view of problems faced by system test and the solutions of our experienced system testers.

The system test patterns are valuable because they document the way we do business. One challenge we face in publishing articles about our patterns is that the best patterns contain stories about real projects. Unfortunately, these stories contain sensitive or proprietary information and can't be shared outside the company. Removing this information usually leaves a thumbnail sketch of a pattern, which may not have much credibility. This publishing restriction is found across the industry. Since most members of the pattern community are not in academia, there is little incentive to publish anything. We at AG Communication Systems are lucky. We are rewarded for external publication. Nonetheless, the publishing dilemma is a problem we face as the patterns community grows and more patterns are uncovered.

Mining Your Own Experience

The extraordinary thing about patterns is the underlying premise that we all have something to share and we all can learn from each other. Mining your own experience is something we can all do. When Jim Coplien and Bob Hanmer taught that first patterns writing class at our company, I had my first opportunity to mine my own experience.

Everyone in the class was anxious to produce something worthwhile, since Jim Coplien, patterns and C++ guru, would read it. Each of us combed our past for some nugget of expertise to document and share with the group. I finally decided to write a brief summary of my Ph.D. work. I had spent years doing the research and the requisite statistical analysis [Rising and Calliss, 1994] to say one simple thing: each software module should contain one design decision. The intent of this pattern is that whatever a module is, function or procedure, package, class, or even an entire sub-system, it should be viewed by a user as providing a single service or capability. It should be possible to state: the module does this! This comprises one component of information hiding as described by David Parnas [1972] . Information hiding is hiding a single design decision behind a minimal interface. The pattern, "Single Design Decision," can be found on our external patterns home page [AGCS].

Mining in Meetings
For the last couple of years, I have noticed that the world is increasingly one big mother lode of patterns! I see patterns in everything I do. This reminds me of learning to type when I found myself mentally typing every conversation. When I learned French and German, I mentally translated every conversation. Sounds like a pattern!

It certainly makes meetings more interesting! I treat every meeting as a pattern mining opportunity. I serve on a team with George Jester, someone who knows our company and the people in it very well. Our team gives many presentations to our vice-president's staff and we have experienced what works and what doesn't when it comes to effective presentations for upper management. During post mortem sessions of some of our not-so-successful presentations, George has provided interesting analyses that I have captured as presentation patterns. Some have been written as exercises in pattern writing sessions and can be seen on the Phoenix Patterns Group Home Page [Phoenix].

I also took potential pattern notes at a team meeting when David Saar, one of our business and marketing leaders, gave a presentation to help our product development team prepare for its customer interaction meetings. AG Communication Systems, like many companies, has chosen the path of self-directed work teams. As a consequence, developers are asked to play a variety of new roles. One is to interact more closely with the customer. These patterns can be seen on our external web site [AGCS].

Some customer interaction patterns appear obvious, for instance, "Mind Your Manners!" Unfortunately, as Jim Coplien has noted: "Common sense is so uncommon!" Many times we are tempted not to document "common sense," thinking that "everyone knows that!" In reality, of course, Coplien is right, not everyone does know that. How many times has a customer trouble report been written for something that everyone should know? Another patterns guru, Kent Beck, has commented [1996]:

"The patterns that I have written with the broadest impact were the ones that I was tempted not to publish because -- everyone knows that."

Mining in Classes
My coach, Tom Snelten, attended a business simulation course where a game, Tango, was played by teams. Business decisions were made and business cycles were played out. The instructor, Brent Snow, had prepared some supplemental materials for our business leaders: "Hiring and Retaining The Best and The Brightest, A Compendium of 25 Strategies and Best Practices." As Tom read through this document, he saw that these strategies and best practices were clearly the beginnings of patterns. On his recommendation, I took the course and spent time after class

talking with the instructor. Brent and I have agreed that we will write a paper that presents these patterns.

Peter Senge's book The Fifth Discipline [1994] describes a collection of system archetypes. One archetype, "Shifting the Burden," addresses the problem of attacking symptoms without understanding the root cause of the problem. As a result, our solutions, while they appear effective, are superficial at best, and may make the problem worse.

I saw this archetype in action in the Tango course. I was part of a team that played the role of a company in simulated business cycles. My team, Gamma, made a decision to develop our personnel competence. In the previous round of the simulation, we had lost a key person to a more capable company and had been unable to follow through on some of our plans. Can't you hear fists pounding in the board room? "We won't let that happen to us again! We're going to meet that challenge head on!" Sad to say, that's exactly what our team or company did. We did meet that challenge. Our competence increased tremendously. Unfortunately, as a result of our focus on increasing competence, Gamma nearly went bankrupt! We weren't even aware of what we had done until the end of the simulation cycle when we found we were barely able to pay our taxes. After that, each year was spent trying to keep our heads above water. We never did fully recover. I learned that a company can borrow on its accounts receivable - good for the short-term but a painful way to do business! If this can happen in one round of a game with three smart people looking at all the parameters, imagine the potential for disaster in a large corporation, state or federal government.

One pattern I mined from the Tango class is "What have you done for me lately?" The essence of this pattern is that a company can't bank points with developers who have high expectations. These "high flyers" want continual growth in competence and, therefore, demand on-going experience on challenging projects. These individuals aren't satisfied with opportunities presented in previous years. They must be challenged each and every year or they will leave the company.

Research Issues and Summary

There's a lot of work to be done in the area of patterns mining. We are at the gateway to uncharted territory. Software developers are having successes with patterns but there is little research to develop solid theory. There are several unexplored but useful topics that call for investigation. As I see it, our most important task is the development of a model for patterns mining and training. Some members of the patterns community feel the best way to capture and use patterns is to employ a categorization scheme and search engine that allows problem solvers to find the right pattern for a given situation.

I have developed my own model for this process, which matches that of other engineering disciplines that have a handbook of best practices. Certified professional engineers in these disciplines are trained in the best practices of the field and apply them when creating products. I think we are a long way from having a well-defined handbook but we can certainly begin to make progress in that direction. As we add to our fledgling handbook, we provide training in the patterns we have at that point and continue to add to the handbook and offer more training.

We also need to know more about how people successfully develop software. This broad topic should include all techniques at every stage of the software lifecycle as well as all the organizational and managerial factors that affect notable achievement. These components will determine the scope of our handbook of best practices.

We must learn from related disciplines. Others have tried to show how people learn, how people solve problems, how components are stored and retrieved. In an age of increasing specialization, we must join hands across academic and practical boundaries to share what we know and expand our viewpoint. As Weinberg has noted [1996], "None of us is as smart as all of us!" This could be the most powerful outcome of the patterns movement - real sharing across disciplines that would enable us all to be better at what we do. What a challenging and hopeful prospect!

References

Adams, M., Coplien, J., Gamoke, R., Hanmer, B., Keeve, F., and Nicodemus, K. 1996. "Fault-Tolerant Telecommunication Patterns," Pattern Languages of Program Design 2, ed. J. Vlissides, N. Kerth, and J.O. Coplien, p. 549-562. Addison-Wesley Publishing Co., Reading MA.

AGCS Patterns Home Page: http://www.agcs.com/patterns

Alexander, C.A. 1979. The Timeless Way of Building, Oxford University Press, New York, NY.

Armstrong, D. 1992. Notes from Managing by Storying Around: A New Method of Leadership, Doubleday Dell Publishing Group, Inc., New York, NY.

Beck, K. 1996, posting to the patterns listserver.

DeLano, D. 1997. "Pattern Mining," to be published in Best Practices: A Patterns Handbook, SIGS Books, Inc., New York, NY.

DeLano, D. and Rising, L. 1997. "A Pattern Language for System Test," to be published in Pattern Languages of Program Design 3, Addison-Wesley Publishing Co., Reading MA.

Gamma, E., Helm, R., Johnson, R. , and Vlissides, J. 1994. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Publishing Co., Reading, MA.

Johnson, R. 1995. Electronic mail exchange with the author.

Johnson, R. 1996. Personal conversation with the author.

McCarthy, J. 1996. Dynamics of Software Development, Microsoft Press, Redmond, WA.

OOPSLA, 1996. "Introducing patterns into the workplace," http://www.agcs.com/OOPSLA/intro.html

Parnas, D. L. 1972. "On the Criteria to be Used in Decomposing Systems into Modules," CACM, 15(12):1053-1058.

Phoenix Patterns Group Home Page: http://www.radsoft.com/patterns/

Rising, L.S. and Calliss, F.W. 1994. "An Information-Hiding Metric," J. Sys. Soft., 26:211-220.

Senge, P.M. 1994. The Fifth Discipline, Currency Doubleday, New York, NY.

Shank, R. C. 1990. Tell Me A Story, Charles Scribner's Sons, New York, NY.

Weinberg, Gerald. 1996. Electronic mail exchange with the author.

Further Information

There's an immense amount of information about patterns (and everything else!) on the World Wide Web. A good place to start is the AG Communication Systems Patterns Home Page: http://www.agcs.com/patterns. There are patterns, published papers, a sample pattern template, information about writers workshops, information about the patterns conference in Arizona, ChiliPLoP, and links to a host of other pattern sites around the world.

The Pattern Languages of Program Design (PLoP) conference is held annually at Allerton House near Champaign, IL. Many patterns from the first three PLoP conferences and the first EuroPLoP conference in 1996 have been published by Addison-Wesley:

Pattern Languages of Program Design, 1995. ed. J.O. Coplien and D. Schmidt, Addison-Wesley Publishing Co., Reading MA.

Pattern Languages of Program Design 2, 1996. ed. J. Vlissides, J.O. Coplien, and N. Kerth, Addison-Wesley Publishing Co., Reading MA.

Pattern Languages of Program Design 3, 1997. To be published, Addison-Wesley Publishing Co., Reading MA.

Information on other patterns books: http://st-www.cs.uiuc.edu/users/patterns/books/

Information on the patterns conferences: PLoP, EuroPLoP, UP, and ChiliPLoP: http://st-www.cs.uiuc.edu/users/patterns/conferences/

OOPSLA (Object-Oriented Programming Systems, Languages, and Applications) is sponsored by ACM SIGPLAN and held annually in October. It is the premiere object-oriented conference, attracting thousands of attendees. There are typically several patterns workshops, tutorials, presentations, and papers.

"Patterns Mining," Chapter 38, Handbook of Object Technology, Saba Zamir, editor, CRC Press, 1999.

Copyright (C) 1999 CRC Press LLC, used with permission.

© 2001 AG Communication Systems, used with permission.