

# Pattern Languages in Practice:

## E-Business Consulting and *The Timeless Way of Building at Destiny*

by Russell Holt

**D**estiny is an e-business consulting firm to large financial institutions. Founded in 1994 amidst the early online banking wave and rise of the Internet, Destiny has grown 50% (to 100 people) in the past year. This article is based on Destiny's foray into the world of architect Christopher Alexander's pattern language concepts, applied to our consulting business.

### MASTERY

A friend was a brilliant cordon bleu chef. From the random contents of even the sparsest refrigerator, he could concoct amazing gourmet dishes because he understood the patterns at a deep level. He didn't consult the *Big Book of Common Refrigerator Food Patterns* and apply a gourmet food development methodology. He simply knew the language and went crazy in the kitchen. This kind of mastery, indeed artistry, is the highest aspiration for an individual and the essence of the pattern language concept as articulated by Alexander. The point is neither productivity nor efficiency, but life, Quality, even spontaneity.

Lao-tzu said, "The Tao's principle is spontaneity." It is freedom to creatively change your direction, despite what you may have planned, when the environment changes, when reality intervenes, or when it just feels

right. It requires the rare ability to see reality untainted by preconception, ideology, the desire for "closure," or comfort in stability. It is profoundly pragmatic. It is often the case that on the Internet, organizations must rapidly adapt their strategies to keep pace with reality. In financial services, rebuilding the business model around the reality of the Internet and its natural patterns is a huge challenge — of perception (what is the reality?), of ingenuity (what interesting things can we do there?), of agility (how quickly can we adapt?). Financial institutions read the analysts' reports and see a future radically outside their comfort zone, far outside the patterns on which they established their modern but pre-Internet business, and so they hire Destiny to do Web strategy, enterprise architecture, and Web project implementation. In parallel. Very fast. Because the world isn't waiting. The different, the innovative, the original, the *spontaneous* wins on the Internet, even in financial services — as long as the solution suits the Internet environment.

Since day one, Destiny had been deeply involved in object-oriented projects, and most developers had a strong interest in design patterns. But although many books on patterns in software mentioned Christopher Alexander's influence, his theoretical ideas were largely abandoned after the introductory remarks. It seemed as if the essence of *The Timeless Way of Building* had been lost in translation. Most software practitioners with an interest in design patterns have not read Alexander's work, but to us, it was inspirational at a time when Destiny was maturing and needed to formalize its consulting practice. We saw an opportunity to try to incorporate an Alexander-like approach, from *The Timeless Way of Building*, into or in place of a methodology. We realized what we'd learned about the Internet and financial

services was a pattern language waiting to be written down. We thought that understanding Destiny's pattern language ought to amount to understanding the fundamental nature of e-business on the Internet, which makes for a far more powerful and innovative company than being experts at efficiently going through the motions of a methodology.

## PATTERNS IN SOFTWARE

It's possible to discuss the role of patterns in the software process without mentioning Alexander or his theories. Popular books and magazine articles on patterns in the software industry generally aren't about a software version of the process of building as described by Alexander in *The Timeless Way of Building*. They are about object-oriented design, or analysis, or using patterns with the UML, or so-called "anti-patterns" (i.e., what not to do). But Alexander focused on the process of building. His 253 patterns in *A Pattern Language* weren't written to simply catalogue architectural knowledge. They work together and are substance for the timeless way. It's great to know that alcoves help a room serve a group better by breaking the room's homogeneity, for example, but Alexander's goal is not a building that exhibits many patterns but *the quality without a name* that can only be achieved through the timeless way of building. It is not obvious that the pattern concept makes sense outside of the theory. So where is "the timeless way of building software"? Does that make even sense? Could the theory apply to sales, or to business strategy? Is quoting Alexander outside his architectural context valid? Surely we can easily write business strategy patterns or software design patterns that are similar in form and intent to Alexander's patterns, but is that enough?

Alexander writes:

The language and the building it creates begins to come to life when I begin to be relaxed about what happens next. I can work in the order of the language, without worrying about the patterns which are coming later, because I am sure that, no matter what happens, I will always be able to find a way of bringing them into the design, when I come to them. I don't need to take precautions in advance.

The language is structured such that the early patterns solve the big problems in ways that do not constrict patterns that come later. Good patterns must be written with this in mind by those who deeply understand this process of building. The process and the patterns are inseparable. One can't escape the conclusion that the timeless way of building is incompatible with most of the staples of the software industry — detailed requirements gathering, for example. Enumerating the details up front is essentially driven by fear: fear of failure, fear of forgetting, fear of change. It's an attempt to insure, to predict, and to estimate, but its basic inflexibility combined with such destructive tendencies as contract-driven change control bureaucracy handily kills the timeless way of building. We can relax if we have the patterns and encourage individual mastery above procedure.

## DESTINY'S PATTERN LANGUAGE

One of the highest-level patterns in Destiny's pattern language is "agile strategy," or continuous reinvention, which essentially states that remaining agile is the most important ability for a competitive offering on the Web. Delivering the right solutions to complex problems on Internet time

---

**One can't escape the conclusion that the timeless way of building is incompatible with most of the staples of the software industry — detailed requirements gathering, for example.**

---

**Agility is more important  
than prediction.**

demands a fundamental integration of business strategy and technology. This means that product strategy cannot precede implementation; the two must evolve together.

This pattern addresses one of the key forces of the Internet: technology as a driving force of business, no longer mere implementation. The world is changing so rapidly that simply doing what we did before only faster is insufficient — rather, we can no longer make plans and strategies the same way. Agility is more important than prediction. The primary goal is to create an environment and a process in which we can constantly revise our strategies, tactics, and technology decisions together based upon new experience, challenges, and opportunities.

With this pattern, Destiny might begin working with a financial institution; it could be the first pattern in the language for our work together. We need not worry about the more detailed problems, because this pattern will create a framework that allows us to attack the rest in a way natural to the Internet — continuous reinvention.

Intuitive or not, it is difficult in practice. This accounts for another pattern, one way to effect the integration of strategy and technology, which is co-location: unite a team otherwise divided between business people and technologists into one location, preferably within a fast-paced, Internet culture. If the team is to use the pattern language process successfully, it must be free to do so. This co-located team should be run as if it were a startup company; that is, given as much autonomy as possible.

Another key force of the Internet is its distributed nature, which allows us to put systems into production frequently, resulting in the “get it out now, fix it later” mantra of Web-based applications. Phrased like that it seems like a vice, but as long as there is a

strong foundation of corporate strategy in place, the ability to release quickly and react can provide true flexibility and be an enabler of the higher-level patterns. An agile strategy requires immediate reaction to quickly available feedback, the best of which comes from the Internet, through the system itself. The shorter the iterations, the more quickly one can react. Iterative development is truly less risky on the Internet than monolithic development, since long, inflexible cycles leave one vulnerable to unpredictable and inevitable changes. We have many patterns that address this in different forms and contexts, such as continuous development, requirements by prototype, staged deployment, and many others.

Alexander writes:

The fundamental philosophy behind pattern languages is that buildings should be uniquely adapted to individual needs and sites; and that the plans of buildings should be rather loose and fluid, in order to accommodate those subtleties.

Customers are individuals, not demographics, and instead of statistical extrapolation of surveys based on focus groups, the Internet lets us converse with each individual to discover his or her actual needs. One example of agility, therefore, is to leave our product plans loose and fluid, or high level, as Alexander suggests, and adapt to the environment we find. Instead of designing in entirety up front, we make our best guess today and get it out as soon as possible. Do a little and see how customers react. Be prepared to alter the entire plan based upon feedback from customers, from the market.

Ultimately it all belongs to the customers. The more they can be involved in the process, the more the outcome will be

uniquely adapted to them, the more valuable it will be to them, the more successful the business will be.

### SUMMARY OF DESTINY'S APPROACH

1. Create an environment to facilitate business agility by integrating product strategy and technology in a single, co-located team.
2. Keep product plans loose and flexible. "Get it out now, fix it later": develop and deploy the strategy and technology in parallel, iteratively.
3. Cycle feedback from the market, from actual users, from project experience at every stage into the design and the strategy.

### STRUCTURES IN DESTINY'S PATTERN LANGUAGE

As of May 2000, Destiny has 72 patterns in its language. Not all are five star, world class, because the language is a work in progress. Many patterns had been in common use at Destiny for years, without our speaking of them as such. Actually writing down the language began with a brainstorm one day in June 1999 that resulted in six patterns; the language has grown slowly in fits and starts since then. Very early on, an intranet application, the pattern server, was created to facilitate the recording and group editing of patterns. It gave the minimum structure of a pattern — its name, author, problem, discussion, and solution — and allowed anyone to add or edit an existing pattern. Patterns could be thrown into multiple categories. One could view the list of all patterns or break them down according to category, or problem, or solution, or various summary views, and so on. Viewing a pattern would list other, related patterns by category, so a project management pattern might hyper-

link to other patterns related to project management. Today, one can explicitly link patterns in arbitrary ways, allowing users to overlay many possibly distinct networks of relationships, exposing higher-level patterns, meta-relationships, and other emergent properties of the whole. We have found several distinct high-level kinds of patterns in our language of less than one hundred patterns; undoubtedly there will be more as it grows.

Although an individual pattern may be given a name, well-formed and concise problem and solution statements, a brilliant discussion of the forces, and suggestions on implementation, several patterns can discuss a common or similar concept at an abstract level. Grouping patterns by these common high-level concepts allows many distinct means of organization; so far we explicitly organize patterns by principle, by client engagement/project, by role, and by general category. The most interesting one is principles.

#### Principles

Reducing risk, being innovative, doing things quickly, being flexible — these are all principles or benefits that we'd like to describe our work. Grouping patterns by principle, and in some cases giving a hierarchy of decreasingly abstract principles, provides an interesting tool to help understand the language and how different patterns relate to these abstract goals. We see relationships among patterns that would not otherwise be obvious. We have many patterns that can be said to be about reducing risk, for example. But separately, independently, one can't tell this from the name of the pattern, and it isn't even always obvious from reading the pattern. It's especially educational when your interpretation of a pattern on these terms is different from another person's.

We often say that patterns collectively reveal principles, and that patterns follow principles. Artifacts of the process, such as the deliverables and things we create, both follow and expose patterns (See Figure 1).

### CONCLUSION

Destiny has been consciously working on a pattern language for roughly a year, though many of the patterns we began writing down had been in common use for a long time, the result of hard-won experience. One of the key reasons for the language's success thus far has been its grass-roots nature: the pattern language intranet application allows anyone to contribute a pattern, to edit patterns online, and to express semantic relationships among them. We saw higher-level patterns emerge from the individual patterns, experience hidden in between the patterns that was not explicitly written down anywhere.

But it's still not easy to understand. We have bits and pieces of more traditional processes that we have put in place from time to time. Ultimately, every actual client project is unique and depends upon the many "environmental" factors we encounter, such as the "personality" of our client, the

technology landscape, the scope of the project, the geographical dispersion of the teams, the market, and many other factors. We have no one-size-fits-all approach except at the very most abstract level — we're going to be innovative, we're going to work quickly, and so on. These principles lead to a variety of individual patterns that can be applied in various circumstances. The primary initial goal to which we aspire in all engagements is to set up for success on the Internet. That means configuring the team and the process we'll follow to be in harmony with the forces at play in the market and the characteristics of the financial institution, its customers, and the Internet in general.

That's not too far from Alexander's theory. So our experience tells us that it is possible to apply Alexander's theory to the software industry, but we believe that it requires the same holistic approach sought by the timeless way. Or at least that is what we think will make it most successful. The pattern language doesn't begin with object analysis and design; it begins with the forces driving the business. For Destiny, Alexander's architect-builder is not the software architect, but the cross-functional team uniting business and technology. The "environment" Alexander frequently refers to — that is, adapting to local conditions — is for us the market, the Internet, even our client.

We have seen early success with co-location and other interesting ways to create unified teams, but the Internet is changing even faster today than it was in Destiny's early days. The critical mass point was passed many Internet generations ago, and radically new business models are constantly emerging — even some from Destiny, we hope. It takes a master who deeply understands the patterns of the domain to understand what works. Destiny is trying to become that master by abstracting our experiences into a pattern language from

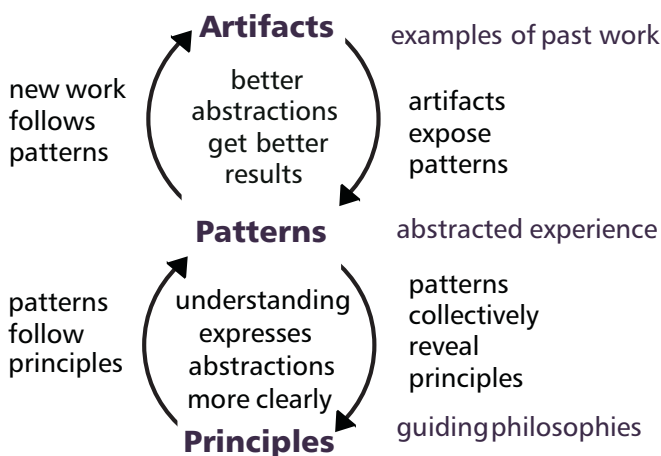


Figure 1 — Relationships among artifacts, patterns, and principles.

which anyone here can benefit and to which anyone at Destiny can make significant contributions.

We would like to take the process closer to Alexander than we have. Doing so will require an entirely new way of working with our clients and new ways of engaging end users. To be true to anything even remotely

resembling the timeless way requires honest flexibility unifying both business decision-making and implementation-execution, unencumbered by bureaucracy. We would ask anyone interested in these ideas to simply read Alexander if he or she hasn't already. As he points out, "A man who is not afraid to die is free to live," which is the story of all startup companies.

**NOTES FROM FIELD RESEARCH: A B-SCHOOL PROFESSOR DISCOVERS DESTINY**

For the past two years, I have been watching with keen interest the evolution of a distinctive way of doing business at Destiny Websolutions. What attracted me to Destiny was its propensity for, and skill in, changing direction. Organizational agility is among the most important characteristics of successful Web upstarts. The company's approach to maintaining this agility, based on the ideas of architect Christopher Alexander, struck me as exciting and innovative. This spring we completed a Harvard Business School case on the company and taught it to MBAs with CEO Lucinda Duncalfe in attendance.

The fundamental issue in the case is one that every startup encounters. As the rapidly growing organization reaches a certain size — around 30 employees, by most accounts — informal, ad hoc methods of doing business stop working. The business won't scale. A common sentiment expressed when this happens goes something like this: "We can't go on like this; we've got to define some processes." But enlightened companies such as Destiny recognize a threat to agility in this sentiment. Close compliance with well-defined processes can be a description of bureaucracy, a particularly deadly ailment for Web startups. Thus arises a fundamental tension for rapid growth companies between the need for consistent, scalable organizational responses and the need to keep those responses changeable, dynamic.

Destiny manages this tension through its use of patterns. Agility is an explicitly nurtured core competency, and there is a vital linkage between the pattern language approach and corporate strategy, which emphasizes shifting target niches to stay ahead of the commoditization curve (thereby maintaining high margins). The resulting performance is externally verifiable. A venture capitalist (VC) interested in Destiny recently called me because he had heard about the case we had written. He noted that Destiny compared extremely favorably with other professional service firms on the metrics VCs use to evaluate such firms.

Some will be inclined to deny the novelty of what Destiny is doing. When we taught the case at Harvard, some students insisted that the company's pattern language intranet was nothing new, just a knowledge management system the likes of which may be found at many consulting firms. CEO Duncalfe's response seemed, to me, just about right. She explained how senior managers hired by Destiny from other consulting firms go through "five stages of grief." First they deny that there is anything new about the pattern approach. Then they grow angry...and so on, to acceptance. Whether or not you follow this pattern (pun intended), I recommend that you read Russell Holt's article closely, and more than once, too.

Rob Austin  
Harvard Business School

*Russell Holt is head of emerging technology at Destiny WebSolutions Inc., where his mission is to help Destiny "architect the future of financial services" on the Web. Mr. Holt has been programming since age 11, and his professional career began in 1995 when he joined Destiny, then a one-person Internet start-up run from the founder's basement. Destiny now employs 100+ people and has delivered Internet consulting and Web solutions to many of the largest financial services brands in the US. At age 28, Mr. Holt was recently selected by Philly Tech Magazine as one of Philadelphia's 30 Under 30 "most exciting people to watch" in high tech.*

*Mr. Holt can be reached at Destiny, 1100 E. Hector Street, Suite 100, Conshohocken, PA 19428, USA. Tel: +1 610 834 0308, ext. 207; E-mail: russell.holt@destiny.com; Web site: <http://www.destiny.com/>.*