Case 4–5

Destiny WebSolutions, Inc.[1]

Destiny's chief executive officer (CEO), Lucinda Duncalfe, lingered in a conference room after meeting with the senior staff to review a proposal that soon would be submitted to a prospective client. She thought Destiny had a good chance of winning the new business even though the small company probably was competing against giants of the consulting industry such as Andersen Consulting and IBM Global Services. Referrals from satisfied customers and a growing reputation were translating into increasing success for Destiny as the end of 1999 approached. We're gaining momentum, Duncalfe thought, smiling. What pleased her most about this was the fact that the new successes had come so quickly after what could have been a traumatic change of direction for the company.

In February, Destiny Software, a product company, had become Destiny WebSolutions, a Web consulting services firm. Duncalfe had pitched the plan to the board of directors in late 1998, and they had agreed that it should be undertaken while the company was operating from a position of strength. The shift represented the company's third change of identity in three years. Destiny's ability to target business niches, grow into them, and then shift quickly to larger targets had become a core competency. This kind of corporate agility was, Duncalfe believed, a major reason for the company's recent successes, but she was the first to admit that frequent changes posed challenges.

One such challenge had been a focus of the just concluded meeting: how to answer the part of the client's request for proposal (RFP) that asked what kind of "methodology" Destiny planned to use in completing the project. Destiny's competitors, especially large consulting firms, had a ready answer to this question. They used methodologies—collections of formally defined standard processes—in all the work they did for clients. Although their methodologies were different, all had common characteristics that were comfortably familiar to clients.

Destiny, in contrast, relied on a collection of "patterns" for structuring company activities. Patterns were not exactly standardized processes, but they were intended to promote some of the advantages of standardization, such as consistency and reliability, while retaining the flexibility to address new and unusual circumstances. The concept of a pattern was derived from the ideas of the building architect Christopher Alexander. Destiny's use of patterns had begun before its transition to a consulting company, and the work on patterns had continued to evolve since that time.

Duncalfe saw pattern-based management as an innovation, potentially a very important one. The company's employees, especially some members of the senior staff, had enthusiastically embraced the concept, developed it, and now seemed to be using it productively; that was behavior she wanted to encourage. Unfortunately, though, using patterns to do business was difficult to explain. Prospective customers were not comfortably familiar with the approach. This mattered now that Destiny was a consulting firm, especially when RFPs asked specific questions about methodologies.

At the meeting, the group had agreed on precise wording to answer the RFP question; they had also agreed about who would explain the answer if questions about it came up in a presentation. But the issue of how to talk about patterns with clients—or with investors, for that matter—was a general one. Although Duncalfe believed in the approach, she had to admit that adopting a more conventional methodology might solve some problems.

As she stood to head back to her office, a variety of questions assailed her. Was the pattern approach really the right way to go? Her employees were spending a good deal of nonbillable time building infrastructure to support this way of doing business. Her best software architect was building a "Pattern Language Server" to act as a repository for the company's patterns. Moreover, Destiny was truly breaking new ground here. There was nowhere to seek information or advice about whether the pattern approach would continue to be useful as the company and its projects grew larger. In the long run, would it be an expensive distraction that might even cause the company to lose business? Or would it be a source of competitive advantage that other consulting companies would have a difficult time replicating?

This case was prepared by professor Robert D. Austin and Doctoral Student George Westerman.

Background

When Lester Shuda founded Destiny in early 1994, he was acting on a long-standing desire to create something interesting and different. His background included two degrees in computer science and two jobs at start-up companies. But throughout those experiences his inclination to set off on his own—to build something—remained strong. In 1990 he took a step toward independence by becoming a successful freelance software consultant. Destiny came into being as an extension to that business; he had little inkling of what the company might ultimately become.

Destiny's First Corporate Transition

In 1995 Shuda began an engagement with the Vanguard Group, a large financial service company in the Philadelphia area (where Destiny was also based), to help build a content area on America Online (AOL). He became the de facto technical lead on the project and began to think of consulting at the intersection of online systems and financial services as a potential focus for Destiny. When the launch on AOL went well, Vanguard began thinking about doing something similar on the Web. Shuda offered a proposal: Destiny would build the required Web system architecture for Vanguard for a discounted price in exchange for the rights to resell the resulting products to other companies. Vanguard declined this arrangement but continued to involve Shuda in architecture design. Despite the rejection of his proposal, Shuda continued to ponder the idea of leveraging his consulting experiences into products.

Destiny's only two employees, Shuda and Russell Holt (who had come on board in late 1994), began actively focusing on product development. Shuda described their process in those early days:

> Russell was building prototypes, and we were kicking around ideas. I would come home, and we would talk about what he was working on. He started building a Web server; then Netscape came on the scene, and it became pretty clear that Web servers weren't the market to pursue. But we had this work we'd done, and we applied for Ben Franklin technology funding with the idea being that we would take the Web server stuff and extend it into a tool kit for building financial services systems.

They gained $25,000 from the Franklin fund and began using that money along with consulting revenues to buy equipment and progress product concepts.

In September 1995 they got an unexpected break. AOL wanted to establish a Bank of America (BofA) content area that would include home banking. Remembering Shuda's work with Vanguard, AOL contacted him and asked him to help BofA. Because of the work Destiny had been doing for the Franklin application, Shuda and Holt felt they had a significant start on what they

would need to do the BofA job. Shuda again proposed retaining the intellectual property rights to product ideas resulting from the work. BofA agreed, subject to the system being up and running by June 1996. With funds from the deal, Destiny ramped up to a team of seven with the addition of five contractors by early 1996. The project, although intense, was successful. Destiny had made its first transition, from a consulting company to a product company. Shuda began to look for investors.

New Leadership

In September 1996, although outward appearances suggested increasing success, internally Destiny was reaching a crisis point. Shuda was becoming overwhelmed. First USA (yet another great client catch) had engaged Destiny to help it develop an interface for online credit card applications. At the same time Shuda felt that he needed to devote full attention to refining the business plan and meeting with the prospective investors who had begun to show interest. It was too much for one person to manage; increasingly he was being drawn away from what he believed he was best at: designing solutions for clients. He turned to his board of directors (then composed of friends and parents) for help in finding a businessperson to join Destiny, someone who could take over the duties associated with growing the company.

As part of the search for business help, Raef Lee, a board member, contacted Lucinda Duncalfe, a marketing executive who was at another information technology (IT) company and had a background in financial services. At first the conversations were about Duncalfe taking a role in marketing for Destiny. After Shuda and the rest of his advisers met her, however, they began to see her as a possible CEO. Her candor and ideas had impressed them, and her qualifications included rapid advancement in the financial services industry and an MBA from Wharton. Extensive discussions between Shuda and Duncalfe convinced both of the similarities in their objectives and worldviews, and in December she became Destiny's CEO.

Duncalfe immediately focused on growth and financing. Consulting jobs were coming in from BofA referrals and some additional funding ($100,000) had been obtained from the Franklin fund, but the situation remained tenuous. People were working very hard at heavily discounted salaries; making payroll was difficult in some months. In May 1997 Duncalfe's work with investors paid off when the company received private venture funding of $1.4 million. The cash influx enabled a move into the company's first real headquarters space. Hiring continued to support the additional business. Notably, Raef Lee was hired as the product vice president. Throughout 1997 the client list grew to include Advanta, GE Capital, and Lucent. By fall 1997, the company had more than 20 employees and was relatively flush with consulting business, mainly in the home banking space.

A Change in Product Focus

Duncalfe realized, though, that there was a problem with home banking. It was attracting a lot of attention and had begun to contain some relatively heavyweight players, such as Edify and Security First, both financially strong and with useful partnerships already in place. Giants Microsoft and Intuit also appeared interested, although their intentions were not yet clear. These facts, combined with lack of progress in selling product licenses, prompted a reexamination of Destiny's mission. Four people—Duncalfe, Lee, and newcomers Mike Kirschner (a product manager) and Reade Frank (marketing)—set aside three weeks for a detailed exercise aimed at determining a new and more viable focus for the company. Duncalfe described the session:

> We worked this incredibly intensive, even by start-up standards, three weeks. And we realized toward the end of this that we had three of the top ten credit card issuers on our client list and that we were the only ones who had any expertise in credit card customer acquisition technologies. The idea of acquiring credit card users online to generate revenues looked like a lot easier ROI argument than the cost reduction arguments we had been using for online servicing systems.

Within a month Destiny shifted its focus entirely from home banking to credit cards. Marketing dollars were diverted, as was the sales team, and press efforts were also turned in that direction. Frank, who had established excellent relationships with Wall Street analysts and editors of financial services industry publications, pushed the story hard, and Destiny was able to differentiate itself from the fray of financial services new business activity.

"Productizing" Destiny's Offerings

The company continued to sell new business to existing customers and continued to build the client list. At the same time, it was trying to make use of what it had learned from its experience in the home banking product business. Shuda explained some of the factors that had led to difficulties:

> We found ourselves bidding against Edify and Security First, and ours was a tool kit approach where we would customize the framework to build a system for the client. It was part software, part service, but we found that most clients just wanted to buy a package, plug it in, and go. But we hadn't packaged it. The product wasn't mature enough. And anyway, we felt that those packages were being commoditized and we needed something much more specialized for better profits. In the credit card space, there weren't as many competitors, and systems that were built there were more customized.

Shuda, Lee, and their development team retained the tool kit approach to support the customization that the credit card space demanded, but they also worked on productizing the components of the tool kit. The results were more clearly defined products. Lee described some of the component products:

> The first product is designed to sit on a businessperson's desk, to allow that person to build a credit card application for the Internet, determine how it should look, with all the validation logic behind it, and to let them roll it into production. A second product is aimed at management—what cards you have out there, what terms and conditions apply to each of them, interest rates—all of that. And the third product gathers data in real time about how people are using the website. When solicitations go out by mail—and there are 3 billion of them per year—the card issuer has no idea how that mailing is received. Well, with our product we know exactly where they went, what pieces they filled in before dropping out, where they got confused, all of which is very interesting to issuers.

Despite intentions to become a product company, Destiny's management realized that the company had not yet fully achieved that status. Over the next year they worked on products and expanded their service offerings into the area of private banking, acquiring Northern Trust as a client. In September they secured additional financing of $2.5 million.

Another Transition

Late in 1998, soon after the funding, Duncalfe began to feel that another change of focus might be in order. Destiny's products had been very successful in allowing the company to move quickly into spaces where consulting and customization were needed, but the technology in the credit card and private banking spaces seemed to be commoditizing, just as the home banking space had before. Competitors were beginning to have offerings available to do much of what had seemed novel in Destiny's offerings of a year before. Duncalfe described her thinking at the time:

> We were ahead on all of our targets, but I was getting a gnawing feeling in my stomach. The big guys, IBM, etc., were coming out with horizontal products that were stripping away our products' value.

EXHIBIT 1
Internet Services Market Forecast ($billions)

Source: Adapted from Yankee Group reports.

	2000	2001	2002	2003	2004
Total IT services	285.7	314.2	345.7	380.4	418.6
Systems integration services	161.8	190.0	223.0	261.7	307.2
Internet integration services	22.7	30.9	42.0	57.1	77.7

The real sustainable advantage the company had, reasoned Duncalfe, was the ability to stay ahead of the commoditization curve, to continue to provide high-margin customized services in spaces where commoditization had not yet arrived. Through increasing interactions with clients up to the CEO level, she had developed a greater appreciation of how challenging the Web world was for established financial institutions, most of which were mired deeply in complex legacy systems and MIS mind-sets. Helping clients adapt to the new economy, helping them understand and fend off threats from new entrants and new business models, looked like a growing source of high-margin business. Duncalfe was hearing encouragement in this direction from client executives: "They were saying, 'What I really love is that you have understanding of and respect for my particular business, but you are also *of the Web.*' "

Acting on this thinking, Duncalfe began working on the case for a full-fledged return to the consulting business, with a much broader focus than the one Destiny had originally pursued. In February 1999 the board agreed to support the transition. Destiny had redefined itself yet again.

IT Consulting and the Internet

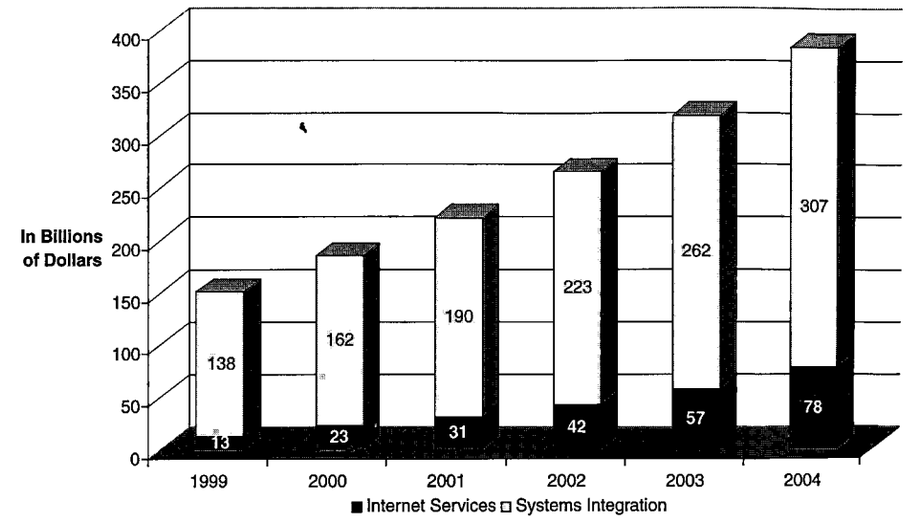
The IT consulting/systems integration industry was worth over $100 billion in 1998 (see Exhibits 1 and 2). The industry was global, but many of the players were regional in scope. Hundreds of firms competed for a share of client revenue and mind-share, and the two top players (Andersen Consulting and IBM) together accounted for only 11 percent of industry revenues.

The IT consulting industry's large traditional players struggled to adapt to the changing IT environment of the late 1990s. The Internet was rapidly making existing skills in mainframe and client-server development obsolete. These firms needed to unlearn old ways and adopt new methods for selling, managing, and delivering projects. For example, large software package implementations were replaced by the delivery of sets of small custom-coded applications. COBOL programming was replaced by C++ and Java. Highly structured analysis and design methods gave way to "learn as you go, fix as you can" approaches. Large teams of specialists gave way to smaller teams of people with interdisciplinary skills.

Another major change was the role of IT in the corporation. IT previously had been considered a back-room support operation. In the early 1990s, with the surge of reengineering-led IT projects and enterprise resource planning (ERP) installations, IT gained new currency as a way of providing competitive advantage. The Internet raised the possibility that IT would become a primary way in which customers and suppliers communicated with a company. Building this kind of e-commerce capability could mean re-

EXHIBIT 2
Projected Demand for Internet Consulting Services in the United States

Source: Adapted from Yankee Group reports.



working parts of a company's business model. As a result, responsibility for e-business projects had begun to migrate from IT to line managers. By 1999 the typical e-business engagement had grown to $1 million to $2 million, with the potential for follow-on business.

An important aspect of the e-business environment was its newness. Clients needed help in understanding the strategic aspects of the Internet. Strategic consulting involved rethinking the business model, helping the client with structural issues, and planning the rollout of the new subsidiaries and businesses. Strategy work provided higher margins and allowed consulting firms to build relationships at the highest levels of their clients' organizations. Tying this to delivery capability provided clients with a one-stop shop for e-business enablement.

Categories of Internet Consultants

Five types of companies provided Internet consulting services:

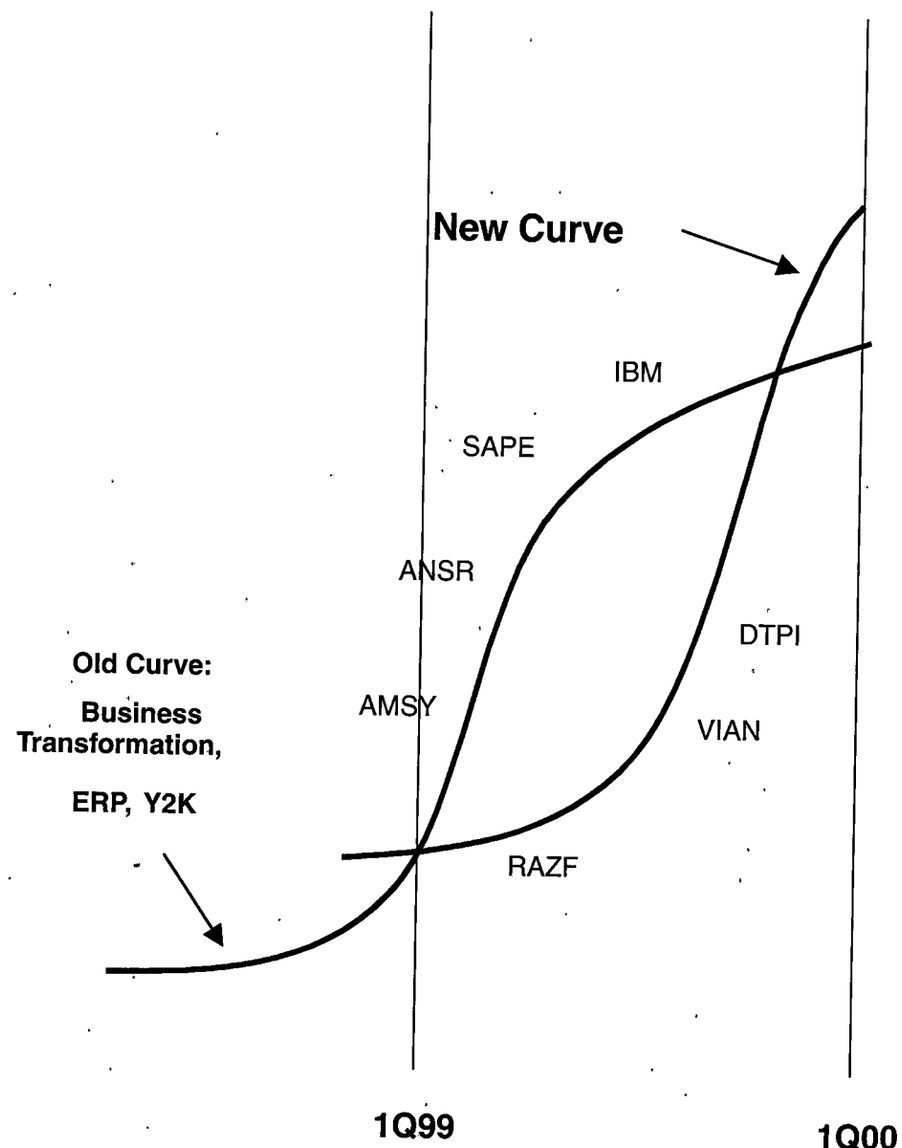
Systems integrators specialized in large system implementation and integration projects. These large firms tended to conduct higher-priced projects where they could leverage their scale. While the category as a whole provided offerings in the entire range of needs, individual firms tended to specialize in either technology or strategy. Examples included Andersen Consulting, Sapient, and IBM. Older firms in this category were working to reinvent themselves by spinning off separate e-business consulting divisions, acquiring smaller firms that had e-business experience, and using aggressive hiring programs to acquire Internet capabilities.

Web design firms tended to be smaller undifferentiated firms that emphasized technical delivery with little focus on business strategy. Most of the estimated 4,500 firms in this category were launched specifically for the delivery of Web-enabled applications.

Interactive agencies entered the industry from advertising. They touted their creative design, branding, and marketing expertise. To these firms, website design and development were a natural extension of the services they already offered to their clients. Notable

EXHIBIT 3
Traditional and Internet Consulting Segments

Source: Adapted from multiple sources.



IBM = IBM; SAPE = Sapient.Corporation; ANSR = AnswerThink; AMSY = American Management Systems; DTPI = Diamond Cluster International; VIAM = Viant Corporation; RAXF = Razorfish, Inc.

firms in this category included Agency.com, Razorfish, and ModemMedia.

Management consultants viewed Web strategy as an extension of their traditional business strategy roles. They could use their powerful brands and traditional high-level relationships to move into the e-business space but rarely went beyond strategy into actual application development Examples included McKinsey and Boston Consulting Group.

Pure Internet players constituted a new segment that crossed boundaries. Firms in this segment, including Diamond Technology Partners, Viant, and Scient, used a combination of creative and strategic talent to develop e-business strategy and messages. They also delivered applications either by playing a general contractor role or by using their own development staff. The firms had different strategic positioning and messages, with some focused on strategy and design while others emphasized their technical skills.

Market Reaction

IT technologies tended to follow S-shaped curves (see Exhibit 3). While traditional services continued to represent a large proportion of total industry revenues, the curve was beginning to flatten. Meanwhile, e-business services, while relatively small, were beginning to take off. Growth in e-business consulting was estimated to be nearly 60 percent per year, while more traditional IT consulting would deliver only 10 percent to 15 percent growth.

By 1999 pure-play consultants and other firms that were "born on the Web" were well ahead of their more traditional peers. But large systems integrators and other traditional consulting firms were rapidly building capability. They were leveraging existing client relationships to build e-business qualifications. They planned to tackle the larger, more complex e-business projects that were just beginning to emerge. In so doing, they hoped to regain share from the upstarts.

The Pattern Language Server

> Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution.
>
> —*Christopher Alexander*[2]

[2]Christopher Alexander, *The Timeless Way of Building* (New York: Oxford University Press, 1979), p. 247.

The concept of a "pattern language" is central to a theory of architecture developed by Christopher Alexander. Since its introduction, Alexander's work has found resonance in fields far from its origins, including, in the last few years, object-oriented programming (a Web search on "pattern language" returns a multitude of entries).

Patterns and Pattern Languages

A pattern, in Alexander's use of the word, is an unchanging and repeating element of a situation or structure. Even though the pattern itself is unchanging, no two specific examples of the pattern are identical, nor should they be. Alexander uses the example of an oak tree to demonstrate how the unchanging can coexist with the unique: All oak trees have recognizable similarities (patterns) in the twistedness of their trunks, the shape of their leaves, and so on, yet no two trees are exactly the same. For Alexander, an object (e.g., a company) is dynamic and alive, able to adapt and extend itself successfully in its surroundings, only if it retains both its unchanging patterns and its tendency to produce uniquely appropriate responses to particular circumstances.

In combination, patterns form languages that can be used by many people. A pattern language is a system of patterns that can combine to produce a rich variety of important outcomes, much as a word-based language can produce a variety of sentences, paragraphs, and books. In an organization, a pattern language might be composed of the meaningful choices that an organization is capable of making. As with expressions in word languages (e.g., sentences), organizational expressions in pattern languages can vary in their quality, that is, their suitability for achieving their intended objective, aesthetic value, and so forth.

Just as people can become more adept at using languages made up of words, organizations can gain familiarity and expertise in the use of their pattern languages. That is, the use of patterns does not require a specialist. However, people vary in their expertise in the use of pattern languages. The example of how master

builders of cathedrals worked is illustrative: "The master builder did not need to force the design of the details down the builders' throats, because the builders themselves knew enough of the shared pattern language to make the details correctly, with their own individual flair."[3] The best way of becoming a master in the use of a pattern language, according to Alexander, is via a master-apprentice relationship.

Destiny's Use of Patterns

When Russell Holt, Destiny's futurist; Paul Rehmet, manager of the WebCycle team; and the chief technology officer, Shuda, explored Alexander's work in depth, they were "totally blown away." They immediately felt that those ideas could solve a pressing problem in their current work. Destiny had grown quickly to a size that was making informal and ad hoc ways of doing business untenable. Company employees had begun to define corporate policies and processes in an effort to bring some order to activities that seemed increasingly chaotic. As they did this, though, a vocal opposition began to worry that too much reliance on policies and processes might constrain innovation in problem solving. Some wondered whether an organization with such a history of dramatic changes of direction should risk locking in on defined processes. As Holt observed, "every system we build is different," and so the organization needed to retain its core competency in customization to fit the circumstances.

Rehmet explained why the pattern language approach seemed like the answer to this quandary:

> A pattern doesn't say, "Here are the 10 steps of the process for doing this." Instead, it identifies the essential elements. It forces the user of a pattern to make his or her own specific decisions. It says, "Here is what people have done before; here are the risks; your understanding of this is key."

Holt decided to try capturing organizational patterns in a Web-based system that he called a "Pattern Language Server" (see Exhibit 4, 5, and 6). "The purpose is not to type in 'How do I do X?'" observed Rehmet. Instead, people were intended to explore and internalize the pattern of past organizational responses and decide, based on these, what the appropriate solution to their own problem was.

An important part of a captured pattern was what Destiny called an artifact. Artifacts were specific examples of things in the past that fit the pattern. As Holt put it, "Artifacts show you what someone else has actually done in a similar situation."

Pattern Relationships

A fundamental idea underlying the Pattern Language Server was that the information needed to solve problems rarely resided in one particular pattern. Rather, problem solving required understanding the relationships between patterns. Rehmet explained this philosophy:

> The goal is for each of these patterns to encapsulate a small amount of knowledge, and to solve problems you must follow the paths between patterns.

Because relationships between patterns were so important, hypertext linking seemed like a natural way to capture problem-solving knowledge, as Holt explained:

> You are able to experience the relationships between things by surfing the hypertext links. You see things in context that way. By surfing through the relationships you see the overall picture better.

The usefulness of the tool was demonstrated when an external consultant joined the Destiny team for a temporary assignment. To get her up to speed on Destiny and the project as quickly as possible, they asked her to spend an entire day surfing the Pattern Language Server. "By the end of the day she totally got it," Rehmet explained. "She totally knew us." Because it had worked so well in that instance, Destiny had begun to consider using the Pattern Language Server as part of the recruiting process. "We want to hire the people who get it," Holt suggested, "and if they don't get it, we probably don't want them."

[3]Ibid., p. 216.

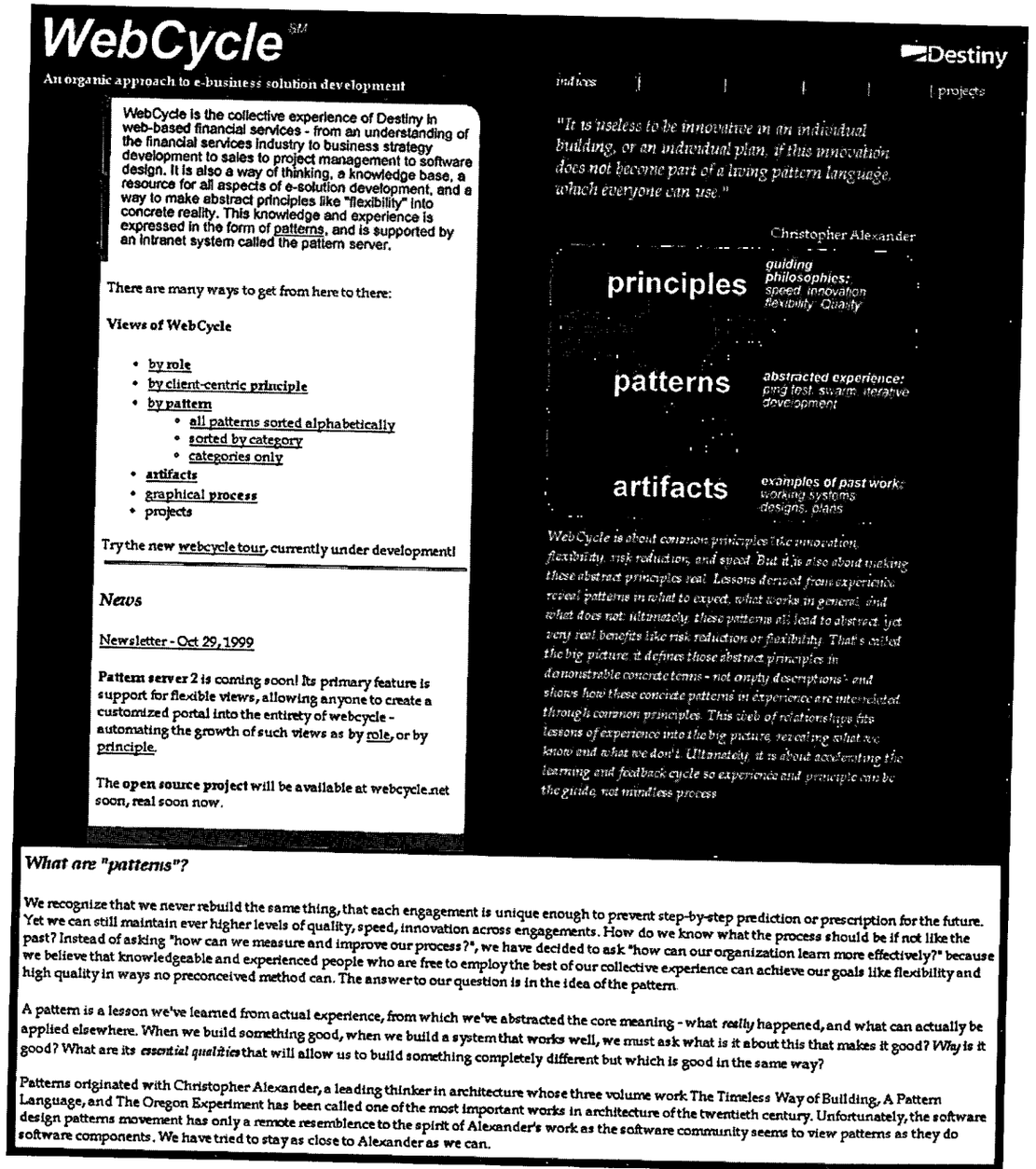


EXHIBIT 4
Pattern Language Server Index Page

Source: Destiny WebSolutions.

EXHIBIT 5 The "WebCycle" Pattern

WebCyclePattern

Problem
Describe the reason, motivation, or "problem" this pattern addresses in one or two sentences. For example, *Long development cycles are not feasible in this fast-changing world. Remaining a leader on the Internet requires keeping up with the latest changes in technology and needs of customers.* Ideally, the statement of the problem should be such that the solution is almost obvious.

Solution
The solution follows directly from the problem and should be written like a rule of thumb. For example, *Short iterative development cycles are most appropriate for Internet system development. Plan development so that new functionality can be deployed after each cycle.*

Discussion

The discussion is the primary content of the pattern. It is a free-form expansion of the problem and solution. Specific examples, pointers to other documentation, diagrams, quotes, hyperlinks, statistics, case studies, roles, etc., are all appropriate here with the caveat that it should be relatively brief and to the point.

The purpose of these patterns is to encapsulate bite-sized knowledge and experience at a specific level and within a specific context—but it is not an exclusive, dogmatic approach. One must choose the appropriate patterns as they make sense in one's circumstance. Considerations for potential patterns:

Be readable. Does the solution follow clearly from the discussion of the problem? Does the problem describe the motivation and set the context for the solution?
Add value. Does the pattern add to the overall experience base?
Be generally useful but not overly specific. Does this pattern describe a somewhat specific situation we expect to recur yet recognize that each circumstance is unique (and therefore not advocate a dogmatic approach)?
Identify relationships to other patterns. Be sure to consider existing patterns that are related and how this pattern might be classified. Most patterns will fit into many categories, just as people are "classified" as mammals and humans at the same time.
It must feel right. Swarm feels right. Doing the right thing in a tough situation with a client, even if it is hard or expensive, feels right. Ping Test feels right. The approach to the problem must be in line with Destiny's values, culture, approach, etc.

WebCycle patterns are modeled after Christopher Alexander's architectural patterns—the father, so to speak, of this way of representing knowledge.

Don't be too abstract . . .
It's important to consider the appropriate level of abstraction for a WebCycle pattern. It may be the case that there are abstract patterns that are applicable at many levels from software design to business strategy, such as "iteration," but the practice and the experience of it is different at all levels. Experiencing iterative software development doesn't prepare one for the iterative development of marketing materials even though the pattern of trying something, getting feedback, and trying again with more information may be the same in both cases. The pattern for marketing materials iteration has a completely different motivation, business context, roles

EXHIBIT 5 The "WebCycle" Pattern *(continued)*

and responsibilities, ways of getting that feedback, etc. It is these differences—the experience and knowledge about the specifics of each circumstance—which make the patterns useful.

. . . or too specific
One shouldn't be overly specific. Continuing the example, an iterative software development pattern is certainly generally useful, but would an iteration pattern for the development of private banking systems add value? It certainly might—if the details of *iterative development* specific to private banking are sufficiently different from any other kind of system.

Patterns Related by Category

Pattern Version Original Author Last Modified by Date Modified hits
WebCycle
Project Launch 1.25 Lester Shuda Lester Shuda 1999/09/24 122 Edit
WebCycle Team Reviews 1.7 Lester Shuda Lester Shuda 1999/09/21 16 Edit
Setting Project Iterations 1.5 Lester Shuda Lester Shuda 1999/09/20 50 Edit
Post-Natal 1.11 unknown Paul Rehmet 1999/09/20 36 Edit
WebCyclePattern 1.12 Russell Holt Russell Holt 1999/08/10 21 Edit
Silo Web Architecture 1.13 Skip Shuda Skip Shuda 1999/09/21 48 Edit

Comments

Add a comment
Your Name:
Comments:

Source: Destiny WebSolutions, Inc.

EXHIBIT 6 The "Swarm" Pattern

Swarm

Problem
A crisis emerges in a project that appears to be a "show stopper."

Solution
Develop a "war room" mentality that matches the urgency of the situation. Swarm the problem with a team of experts, open a frequent and detailed communication path for the client (e.g., daily or twice daily) and execute on a well-thought-out plan.

Discussion
When a project-threatening crisis occurs, it is critical that it be addressed expeditiously and with clarity. Inform senior management internally AND at the client of the problem and the plan to use the swarm tactic (i.e. "we are assembling a team of experts and executing a well-thought-out plan, quickly, to resolve this").

Quickly assemble a team of experts from within the company; be prepared to pull people on the project off other tasks in a support mode. Pull out all the stops—and keep outside expertise open as an option (tools, consultants, vendors, etc.).

(continued)

EXHIBIT 6 The "Swarm" Pattern *(continued)*

Get the most knowledgeable people to describe the problem in detail and try to quickly assess your options. Understand what you don't know.

Organize a multiprong approach which is geared to:

a. Gathering the information that you don't have (i.e., get to know what you don't know but what you should know).
b. Some initial attacks at solutions based on best guesses as to the problem.
c. Put any outside resources that you may need to call upon on alert.

Plan to check in every few hours on progress.

As the team attacks along a and b, have the "war room" manager communicate with the client and senior management on the steps being taken and the next check-in point. Execute on c.

Iterate until the problem is resolved.

The use of Swarm provides a great opportunity for learning. When you are done, you should always ask the question "How can we avoid this in the future?"

ARTIFACTS:

Sample Swarm Communication from the [Major Client Name] project.
Client's response to Swarm Sample from the [Major Client Name] project.

Comments

Add a comment
Your Name:
Comments:

Source: Case writer, as received from company.

Top-Down versus Bottom-Up

Destiny's pattern language enthusiasts believed that the approach combined the best aspects of bottom-up and top-down methods. On the one hand, patterns were not imposed from above. The collection of patterns was, as Rehmet put it, "organic," intended to grow from within the organization. Holt described some of his hopes for bottom-up activities:

> Anybody can edit a pattern; anybody can add a pattern. I'd like to see patterns merge, and split, and change. I'd like to allow people to set up their own little pattern domains full of the relationships that are important to them.

On the other hand, there was a WebCycle team that reviewed patterns to decide whether to promote specific patterns. There were heuristics for deciding whether a pattern was a good one (see Exhibit 7). "One of the team's jobs," said Rehmet, "is to look at a pattern someone has created and say, 'This could be more general.' "

Masters and Apprentices

The pattern approach at Destiny also included Alexander's notion that mastery of a pattern language was best acquired through a master-apprentice relationship. Implicit in this notion is that value creation cannot be completely separated from the people who are expert at the use of patterns. The aim of the Pattern Language Server therefore was not to extract and codify firm knowledge so that the company would be less dependent on people. Dependence on people with high and growing expertise was a fact of life in Destiny's world. The Pattern Language

EXHIBIT 7 Russell Holt's Essay on Organic Software Development

Approaching Organic Software Development

Rigid software development processes and methodologies can work in a static, well-defined space. But when the environment is changing rapidly, when the trail is being blazed, when each project is different, a new approach is needed to avoid exerting as much or more effort maintaining the process itself as doing actual development.

Unlike a person, a methodology or process lacks a mind. It cannot adapt to changing circumstances. We must know when to add to and when to omit from the process, and sometimes when to toss the entire thing. These kinds of decisions cannot be coded into a process—and any attempts to do so are the source of fabled gargantuan Methodology tomes, which can be larger than any project for which they are intended, and which most people simply ignore. They are attempts to substitute structure for intelligence, process for responsibility. This extreme is fundamentally unable to encourage personal growth and mastery, which is strange in an industry where productivity and "quality" can vary among developers by orders of magnitude.

Today, our environment is changing more rapidly than the times and conditions that gave rise to software methodologies. We still have many competing approaches, despite the fanfare the fashionable Unified Process has received. The Structured Analysis approach was similarly popular not too long ago. Still, today as always, each performance on the cutting edge is unique. There is no process for innovation by definition, where, to be successful the challenge is not to find the right process to substitute for intelligent mastery, but to find and grow masters. Masters can do what no process can—something that hasn't been done before.

Master developers who are free to make the most of their experience in an organic, ad-hoc way do the most efficient, appropriate, and error-free work. Masters understand that nothing stays the same, that everything is changing (even though only slightly, sometimes), that no two circumstances are the same. A master typically has the confidence to work without all the answers, because the answers will come as needed—and it's often the case that they cannot appear earlier. Only a novice must wait until all the answers are known.

I think this is how effective small teams operate by default—ad-hoc. It is easy to communicate and express ideas and react quickly to changing circumstances as an individual or on a small team. For this to work in a large team or organization, a very efficient communications infrastructure is essential, with a way of sharing experience at all levels. In addition, experience and knowledge must be represented in a way that allows related things to be linked, patterns of the whole of experience to emerge.

What novices need to grow into masters are knowledge, experience, and the freedom and opportunities to grow. Process in software development suppresses all of these. It is helped along most effectively in a master-apprentice relationship, as stated by Christopher Alexander: "The fundamental learning situation is one in which a person learns by helping someone who really knows what he is doing" and that we should "treat every piece of work as an opportunity for learning."*

One approach to knowledge is its representation in the form of patterns, as described by Christopher Alexander in the books *the Timeless Way of Building* and *A Pattern Language.* A pattern is a lesson, a rule of thumb, abstracted experience, something we've noticed is true (or at least effective). A pattern is not a recipe or step-wise process.

*"Master and Apprentices" (83), in *A Pattern Language,* C. Alexander, Oxford University Press, New York, 1976.

(continued)

EXHIBIT 7 Russell Holt's Essay on Organic Software Development *(continued)*

A base of experience in the form of patterns is very useful, but its use is limited by the form it takes—a book, a library of books, and the WWW are all different. A system—a pattern server—which allows one to see relationships between patterns that are related in various contexts is key to finding the information one needs in time. Searching for relevant information in a book requires that one read the whole book, or that the categorization defined by the index and the table of contents match that which is in one's head. The web is a much more appropriate medium for patterns and their relationships.

Such a system will greatly accelerate learning because it is approachable, it can be perpetually up to date, it is able to be reorganized in a dynamic way, and it can foster grass-roots contribution. Since the web can be a two-way medium, everyone should be able to contribute to it at any time. This reflects the ever changing nature of our world today.

Every organization's experience is different, just as every organization's culture is different. The development, or growth, of the company's experience base requires grass-roots participation. It can only work when the organization's population contributes to the whole, because the most directly relevant experience to an organization is the individuals that comprise it.

Therefore:

Record experience in interrelated, bite-sized rules of thumb, or patterns. Create a communications infrastructure, which allows one to view and visualize the interrelationships among the patterns. Let it grow organically by allowing everyone to add to this experience base. Encourage the personal growth of individuals into masters as a central feature of the organization.

Source: Destiny WebSolutions.

Server, then, was designed to facilitate the development of pattern use expertise in individuals. Holt put it this way:

> Documenting patterns, although useful, is only part of the process of developing mastery. Alexander says that the fundamental learning situation is one in which a person learns by helping someone who really knows what he is doing. This is the way to grow masters in our organization.

Into the Future

Duncalfe was not at all sure where the pattern language work being done at Destiny would take them. She suspected that it would be difficult for other companies to copy. "You kind of have to get it to do this right," she observed, "and most companies won't."

All the people at Destiny were quick to admit that they were still learning about the approach and how to use it best. Rehmet noted:

> We don't fully understand what we're doing with this yet. There's an extent to which we don't know what shape this thing will eventually have.

Holt summarized most employees' attitudes toward the Pattern Language Server:

> There's a lot of interest. Most people see it as a knowledge management system. I think that's okay. What it really is, though, is kind of a corporate lore system. It tells our stories, how Destiny does things. It's our philosophies and practices made tangible.

Holt too had some misgivings. He summarized some of the concerns that were sometimes expressed by others about the pattern language approach and conceded their legitimacy:

> One thing I suspect is missing . . . this is knowledge, interrelated experience, and so on, but it is no kind of substitute for written-down instructions on how to do something specific. I wonder if we might ultimately need a combination of patterns and conventional processes to run a growing business.