Do-It-Yourself IT

BY DOC SEARLS

Doc Searls is one of the ultimate do-it-yourself guys. He does things his way. Among the things he has done are to co-found Hodskins Simone and Searls, a Silicon Valley PR/advertising firm, and to co-author The Cluetrain Manifesto, a book that helped make clues (and cluelessness) famous. He has enlivened many otherwise earnest conferences with his wit and insight, both on the stage and in the bars. Currently, he is senior editor for Linux Journal and a renowned expert on open source and its culture. In this essay he explores a semi-related, semi-orthogonal phenomenon, do-it-yourself IT.

– Esther Dyson

Rewind history a quarter century, to the early ‘80s. What we now call IT (information technology) was still called MIS (management information systems). Outside corporate walls, personal computing was an interesting hobby. Inside corporate walls, it was an oxymoron. Computers that mattered were mainframes and minicomputers, which lived in ventilated rooms on raised floors, tended by a priesthood of professionals. The working synonym for computing was “data processing.” Workstations were called “terminals.” While terminals came in “smart” and “dumb” breeds, none were personal.

Vendors dominated and defined enterprise computing. An MIS department was an IBM, Digital Equipment Corporation, Control Data, Honeywell, Bull or NCR “shop.” It used not only central-processing hardware from those vendors, but also disk drives, terminals, networking hardware, cabling and nearly everything else. Even the third-party “compatible” peripherals in hardware and software were generally designed to work only with one vendor’s breed of computer system – its “platform.”
In addition, just as each vendor’s system was siloed, the systems themselves were isolated from each other – even systems of the same kind. Digital and Wang networks eventually “standardized” on Ethernet, but with incompatible systems called DECnet and Wang-Net. Meanwhile, IBM pushed Token Ring, an incompatible standard that also required IBM’s then-new Structured Wiring System, which differed from the company’s own existing (and widely differing) networking and cabling systems. Interoperability was a rare achievement, and mostly a distant dream.

Looking back, it’s easy to see that customers have always wanted compatibility and interoperability between platforms, along with extensibility of the platforms. They wanted choices at every level, and freedom from dependencies up and down each vendor’s “stack” that limited choices to the vendor’s own offerings. In those old days, customers didn’t get any of that. That’s starting to change.

So the history of IT has been a slow war of liberation – a struggle toward independence for users as well as for professionals.

Each of our buzz-concepts, from object-oriented to client-server to open source to Web services and service-oriented architectures, has been a step on the path to independence – one in which increasingly modular, transparent and easily manipulated components let people build the systems they want.

In a continually self-reinforcing virtuous cycle, the Internet has played a part. It gave people and systems the ability to communicate across silos, and thereby gave them the impetus to communicate better – with systems that were intelligible to and manipulable by other people and systems. Rather than design applications, the professionals in the priesthood will eventually build modules and tools that users can assemble and manipulate...not to build applications, but to perform tasks and processes.

In the construction industry, this kind of independent and self-reliant activity is called DIY, for do-it-yourself. For the computer industry, we’ll call it DIY-IT. For now, DIY-IT is about the empowerment of professionals, not of end-users...though at the applica-
tion level users are demanding and getting applications that are far more customiz-
able – and forgiving – than they ever were before. In our upcoming issue on “beyond e-mail,” we’ll talk more about where this trend is taking us.

As that begins to happen, value creation will follow flexibility and transparency up the stack, climbing inexorably from the specialists toward end-users. We’re about halfway along this road. Right now, DIY-IT is mostly at the middleware level, among those who are technical enough to assess the capabilities and the quality of the code and other available materials they are using, embellishing and extending.

In summary, this issue is about the trend away from monolithic, impenetrable systems to systems that users design for themselves. As with any trend caught early, there are a variety of factors and strategies in play. We cover some of them in the pages ahead.

**Start with building materials**
The move to DIY-IT comes in part from the availability of much better tools and materials. In the old days, metaphorically, do-it-yourself meant chopping down your own trees. Neighbors would join in and help – on the unspoken assumption that when, say, their sons got married, those same beneficiaries would help the neighbors’ sons in turn. As the industry matured, people would hire professionals to do the chopping and the building. But now that pre-cut lumber is easily available, people can do their own building again...or at least expand their own garages and add their own sunroofs.

“Software is so modular today that it’s easy for users to build things. Today’s grown-ups got their programming experience from schools and books. Their starting point is further along than it was for us,” says Marten Mickos, CEO of MySQL. “They are more capable of doing it themselves.”

One reason for DIY is that support is so easily available now, he says. “If you as a user create an application of your own and you put it on the Net, you can get help from anywhere in the world. And if you want to share someone else’s work, whatever your problem, you can find someone somewhere who has solved it.”

However, he doesn’t see a profusion of applications into the public domain. It’s not so much that the interest isn’t there, but the information isn’t there. Even SourceForge, with 80,000 open-source projects listed, doesn’t provide sufficient
information to guide a typical user in making choices. “You can look at the number of people involved in the mailing lists, the quality of comments, the ratio of users to developers,” says Steve Howe of investment bank Dresdner Kleinwort Wasserstein (DrKW). But that’s pretty limited information, and it doesn’t tell you what’s right for you. Those products are competing with companies who are spending multimillions “explaining” their products.

“I discussed this issue recently with someone who suggested we set up an exchange for MySQL-based applications,” says Mickos, “but we just don’t have the bandwidth. To make it successful, we really need people to go out and build it and look for the applications to put in it. It’s not a technical problem, but more a question of attention and presentation.”

In short, people don’t know what’s available, and it’s hard to tell the good from the bad. When something goes wrong, whom can you call?

**Don’t you mean “open source?”**

There is a lot of overlap between open source and DIY-IT, and many similarities between the two movements, especially in their culture. The practitioners of both tend to be individuals bypassing the monolithic systems of old. Both are concerned with minimizing costs and complications. Both are resourceful, and value modular and re-usable designs. Both are far more concerned with use value than with sale value. And both have the same attitude of ornery independence: I do it my way (DIY) and we do it our way (open source).

There are also some big differences – although a single person may fit in both camps and have both sets of motivations. Open source developers care primarily about the methods behind the products they make, and about licensing terms that allow those tools and products continue to evolve. DIY-IT professionals care primarily about getting work done. For them, open-source products are handy and useful IT building material ranging in price from cheap to free. DIY-IT workers also like the way open-source products serve as abundant, transparent and adaptable raw material – the equivalent in code of wood, stone and cement. As Microsoft Windows Server product manager Jackson Shaw once said, “It isn’t that Linux grows on trees. It is trees.”

At this point, open source is an important but coincident part of the DIY landscape. The important feature of open source in a DIY context is not its licensing terms (there are many variants), or the ways the source code can be altered or improved,
but the fact that it can be inspected. Source code has no secrets. That’s what makes it resemble a natural building material like wood or stone, rather than something pre-fab of unknown composition.

A few years ago, a programmer working on a large e-commerce site said, “When I’m building a skyscraper, I want to know there’s rebar in the concrete. With Linux, I know. With Microsoft, I don’t. In fact, NT’s memory leaks prove to me there isn’t rebar in there. Since I have to work with NT for political reasons, I just cope with it. But I know if we could see the source, we could probably fix the problem pretty fast.”

Another big source of DIY building material is open and proprietary tools such as MySQL, which owns its code yet releases it to the world under the Free Software Foundation’s General Public License as well as its own commercial license. In this case, the code is not so much inspectable as manipulable: Its inner workings are hidden, but its interfaces are open and clear – prefabricated with assembly instructions, so to speak, rather than transparent. As DIY moves up the stack, this approach rather than transparent open source may become more prevalent.

In the future, indeed, much do-it-yourself work will also be built on top of products with restrictive licenses – even when there is open and free source code at the base level. We already see this with Apple’s OS X stack, sitting on top of FreeBSD – and with countless devices (e.g. TiVo and other PVRs) running on embedded Linux. This is inevitable in a world where one important strand of computing will be the tracking and administration of intellectual property rights. Fortunately, the very powers of increasingly good software will support lots of choices among products and digital content – including those with and without IP restrictions.

For most of the history of computing, a company’s choices to build or buy – an important aspect of a healthy industry – were severely limited because the technology was obscure and allowed vendors to impose de facto lock-in and other restrictions. Both open source and DIY-IT are ways the IT market’s demand side is starting to supply itself. The era of vendor-limited choice is coming to an end.
Construction: A Grown-up Market Model

The best way to put DIY-IT in perspective is to look at the IT marketplace as a collection of professional and topical specialties, comparable in abundance and range to those of construction. Where construction has roofing, HVAC (heating, ventilation and air conditioning), cabinetry, plumbing, electrical, tile work, framing and demolition (among many others), IT has programming, system administration, database administration, database design, field engineering, architecture and networking (among many others). From this angle IT doesn’t look much different from any other market. But this is not a perspective most of us take. Instead our mental defaults are still stuck in the old vendor-dominated model of the market. This is changing, but it’s not happening overnight.

This change in perception is tied to the relative (im)maturity of the computer industry; when it finishes growing up, it will look much like the construction industry.

There are clues in our vocabulary. We “architect,” “design” and “build” software with “tools.” Among our constructions are “platforms” and “containers.” For the Web, we borrow the language of real estate, building “sites” at “locations” with “addresses.” If a site isn’t finished, we say it is “under construction.”

The construction market ecology is large enough to include many different product development and business models. For example, open source isn’t a cause or a movement in construction; it’s simply taken for granted. If a worker finds a better way to tile a roof or hang a door, he usually shares that knowledge. And given that construction workers move around a lot from job to job, the knowledge spreads easily from project to project and company to company. This is how construction trades constantly change their methods and improve their work. Yet product manufacturers and service providers are free to keep their methods and inventions secret, and to sell products and services based on those secrets.

Outside of natural materials such as wood and stone, just about every building material you can buy is either patented, “patent pending” or subject to some kind of patent claim. Yet patents are a non-issue in construction because a builder is not subject to intellectual property claims by any maker of building material. He can frame a house with pine boards, plywood, screws and nails that don’t have a patent among them. Then he can fill the walls with patented insulating materials from a variety of manufacturers such as Dow-Corning and Johns-Mansville. Finally, he can
use windows from Andersen or Pella. . .or from a less-familiar manufacturer that saves money by not marketing or by making inferior windows.

In addition, building materials are interoperable. (In fact, they are so interoperable that use of the term applied to construction sounds silly, as if one were to point out that buildings are physical. But imagine a construction industry built around a large-scale version of Lego’s IP-protected building blocks, and you can imagine yourself back in the IT world.) No construction materials manufacturer is in a position to insist that contractors construct a whole building on that manufacturer’s “platform.” Framing a house with wood from Weyerhaeuser does not require everything else in the house to comply with Weyerhaeuser’s private standards, or for the customer to pay license fees to the company. (Nor does Weyerhaeuser require replacement of the house’s framing every four years, when the company upgrades its formulation for wood.)

In short, construction-industry commodities are not considered a degraded end-state for formerly innovative product types. Instead, they provide a necessary abundance of choices. Providers find plenty of ways to make money and add value, differentiating by materials, quality control, responsiveness, customer support, warranties or any of the other positive variables that contribute to brand value. Milwaukee Electric Tool Corporation, for example, has a premium brand reputation, earned by making the heaviest-duty tools for professional builders.

The construction industry model is different from IT in one fundamental way: Its commodity materials, such as gravel or wood, still cost money; they cannot be duplicated for free. Commodity IT, once created, can be replicated infinitely and easily practically for free. Yet information still has value at the commodity level. Says Linux Journal editor-in-chief Don Marti, “Information doesn’t want to be free. Information wants to be $6.95.” Apple’s iTunes music store charges – and gets – 99 cents each for songs that also can be downloaded for free over a peer-to-peer service, because iTunes adds value: packaging, delivery, convenience and other features. As with construction, there are lots of ways to make money by putting commodity building materials to use. Still, many IT vendors don’t sell materials, and struggle constantly to decide what to give away and what to charge for.

In the IT market, as in construction, there will always be a variety of different players. According to McGraw Hill Dodge, $3.5 trillion was spent on construction in 2002, making it the largest industry in the world. Naturally, the construction industry includes some extremely large companies, such as Weyerhaeuser, with $20 billion
in sales last year, and Fluor Corporation, at $8.8 billion. But it has no equivalents of IBM or Microsoft. According to WITSA Global Research, the global high-tech industry stood at $2.4 trillion in 2001, and its largest companies, IBM at $89.1 billion in 2003 sales and Microsoft at $9.1 billion. Moreover, unlike the construction-industry companies, their influence extends beyond their revenues.

DIY, over time, reduces that influence. As the value moves from the components to the design and assembly of applications and systems using those components, as is the case in construction already, the value may be created in-house in companies where management trusts its own people and those people are up to the challenge. Or it may be created by vendors such as Accenture and McKinsey (though we may be heading to a world of commodity strategies). As Nicholas Carr says (PAGE 14), the real value — and competitive advantage — lies in what a company designs (and implements) that is unique.

The IT vendors that do sell materials – IBM and HP for example — are finding those markets challenging, and crowded by proud-to-be-a-commodity suppliers such as Dell. They clearly don’t see DIY as a great threat, and they are encouraging open-source software: They see a large market of companies who still don’t want to do it themselves and who will turn to service suppliers such as IBM Global Services ($42.6 billion in 2003 revenues, or 0.5 percent of global IT spending) and HP’s services arm ($12.3 billion in 2003 revenues). Independents such as Accenture and CapGemini are also flourishing — though all of them will tell you that they provide strategic advice as well as implementation services (which you could do yourself or send offshore).

Of course, the construction industry is far from ideal in many ways. It’s full of regulation, slowed by government approval gauntlets, hamstrung by unions, rife with lawsuits, and corrupt in many places. Anybody who has ever built or renovated a house knows that most builders are not like friendly home-improvement guru Bob Vila. Too often, they employ a crew of drunks, ex-felons and misfits who show up on alternate Tuesdays.
DIY-101

Robert M. Lefkowitz, chief technical architect and VP, information technology at AT&T Wireless, is a great believer in DIY – and in NEA, which he credits for the success of open source (as well as of the Internet, another big open-source project).

NEA is an acronym for three virtues:

- Nobody owns it.
- Everybody can use it.
- Anybody can improve it.

To these Lefkowitz adds two more: “Significant business value associated with the use of the software does not reside in the software itself.” And, he says, “It’s kewl.” (The italics and spelling are both his.) Without these five elements, he says, there would be no open source in business.

Yet Lefkowitz, a big fan of both open source and DIY, sees limits to the prospects and the utility of both – at least in the world of large enterprise systems, where he has worked for many years. (Before joining AT&T Wireless, he worked in IT on Wall Street.) He observes that as you move into areas of business operation where a vast amount of highly coordinated person-hours of arcane work is required to develop an application that operates internally, the chance of an open-source project assembling itself through volunteers becomes remote.

He set out to discover the dynamics of open-source development in enterprise software at O’Reilly Open Source Convention last summer, when he gave a talk titled “six missing open-source projects” and challenged a packed room to get cracking on them: billing, CRM, visual programming, business process integration, messaging and data warehousing. There are many others, he said, but we could at least start with those six.

His hypothesis was that to be successful, an open-source project needs involvement from individuals who think it’s kewl, meaning it’s of practical interest to an individual. “Companies are always concerned about people ‘stealing’ their IP,” he says. “[But] in fact, if you just put [your IP] on the table, nobody would pick it up, because most things of this nature [enterprise applications] are not valuable to anyone else without them having to put in a lot of work.”
For example, he says, an individual would be motivated to tinker with word processing software such as OpenOffice or Web-browsing software such as Mozilla for his own use and benefit. The same probably can’t be said for, say, a billing system for a wireless company.

He decided to test his theory at the Open Source Business Conference this spring. During another talk on missing open-source projects he offered a specific challenge: creating an open-source billing system based on the source code of Flexcell, the billing system that had been used by Vanguard Wireless, which AT&T Wireless acquired in 1999. Tongue planted in cheek, he held up two CDs containing Flexcell’s code and explained what was kewl about it, starting with micropayments. Nothing does micropayments better than a telco billing system, he explained. Think of all the small sums of seconds and minutes, and the conditionalities involved — all being tracked in real time for huge numbers of customers on the move.

“So the offer I’m making,” he said, “if anybody is interested, is doing the world’s largest and kewl-est micropayment system. I’ve got the source code. [And] I do have authorization to look into how to open-source such a thing. I’ll be happy to take any licenses or whatnot. It’s all open for discussion.” He went on to pitch other potential big enterprise open-source projects: “CRM systems are really kewl, too.” He even detailed a peer-to-peer CRM system of his own design, called Carester.

In the several months that have passed since then, Lefkowitz hasn’t seen much take-up of his challenges. Why? First, the practical: Jamie Zawinsky, the veteran former Netscape programmer who named Mozilla (among other achievements), famously said, “Linux is only free if your time has no value.” Lefkowitz explains, “The big problem with any open-source project is pushing it out the door. I remember the
first time I heard [open-source guru] Bruce Perens talking about this at HP. They set up a whole group to go through the code and to scrub it to make sure it didn’t have comments; they had to get the lawyers to look at it and make sure they had removed any patent infringements. All that stuff takes a lot of work. If you’re not in the software business, is it worth all that effort? IBM and HP can say ‘It’s part of our strategy. We have to invest in this.’ But for AT&T Wireless, it’s ‘What’s in it for us?’ So open-sourcing a billing system is very difficult. It requires a lot of expertise, a lot of hardware, a lot of time.”

He adds: “I remember a cartoon, with a scientist at a blackboard covered with formulas. In a central section of the blackboard, all these arrows come together. The caption says, ‘This is where Einstein discovers that time really is money.’ In a large enterprise, time really is money. So doing it yourself or having somebody else do it is a cost-of-labor exercise.”

Consider the market served by Home Depot, Home Base and other big-box DIY construction supply stores. They not only serve individuals remodeling or rebuilding attics, garages and homes, but general contractors and subcontractors serving the home-building marketplace. There are no retailers called “High-Rise Depot” or “Factory Base.” Building and maintaining large corporate structures is a different breed of activity. Lefkowitz explains, “At a high level in an organization, there is no DIY. You’re telling other people to do it. Do you hire employees to do it, or hire contractors to do it, or hire vendors to do it? That’s your choice.

The second reason for the lack of interest in his billing system, Lefkowitz explains, is the motivation behind DIY projects: “A cellular [phone company] billing system can only be of practical interest to companies, not individuals,” he says. “Nobody sees how this aids their career. Why should they spend their precious time working on this stuff?”

This brings up the major distinction between DIY and open source in the back ends of large enterprises. DIY is personal. There may be a collective self, of course, in DIY-IT. But DIY is inherently small-scale. It involves crews and teams working on projects, often in an as-needed or ad hoc way, often when there are no alternatives, commercial or otherwise.
OK, so when do I DIY-IT?

DIY-IT is easiest when a motivated individual or group has access to appropriate, transparent tools that make the goal easy to accomplish. This is where open-source tools come in handy.

An individual may practice DIY-IT when she needs to bypass organizational authority in order to get something done. She’ll often use open-source tools, which cost little or nothing and are freely available, and her expertise to glue it all together.

A *company* may also practice DIY either to bypass an authority of sorts – that of a vendor with an inflexible, “full solution” tool – or when the solution it needs does not yet exist. In this case, it often costs more (in time and money) to glue together a solution from existing pieces rather than building it made-to-order and from scratch. Says Google CTO Wayne Rosing, “As we look at it, staging the integration glue to tie together standard third-party packages is often a larger project than building a basic Web-based system focused on the problem at hand. Obviously our internal systems are not bell-and-whistle-rich, and sometimes are too simplistic, but they can evolve quickly and adapt to match the business. So we don’t waste time or energy on stuff we never use. We can flush a bad idea quickly and move on to a better solution.”

Of course, there is more to consider than development and time costs, he points out: “The downside is some third-party systems encode a lot of regulatory and legal knowledge (HR systems come to mind) which we do not have in-house. So there is no one path to follow. The art is balancing. [That’s why] we use Oracle for finance and a few other specialized third-party applications. [But] most of the ‘real-work’ is handled by home-built software. There was no place to go to buy AdWords!” Ofoto is a prime example of a DIY shop, blending DIY and packaged software solutions in its IT strategy.

**Ofoto: The compleat DIY-IT customer**

Ofoto is a consumer digital photography service that provides ways for customers to store, edit, share and print photos. Launched in 1999 by president Lisa Gansky and growing steadily ever since, Ofoto is the leading company in its category – a status certified when Eastman Kodak acquired the company in 2001. (*see Release 1.0, March 2004.*) It continues to operate as an independent subsidiary.

Adam Hertz, VP of technology strategy, describes Ofoto as “a complete DIY shop” – by which he means an IT environment with a combination of custom (DIY), semi-
The DIY movement in IT seems a good example of a market getting smarter faster than the vendors that supply it.

Cluetrain's first thesis was “Markets are conversations.” Markets were real places long before the word “market” was borrowed to label a raft of other things (demographics, regions, categories, bulls, bears, invisible hands. . .). Removed from the grounding clarity of the real marketplace, it became too easy to conceive of business as logistics, which is what we do when we talk about “content” that we “load” into a “channel” for “delivery” to an “end user” or a “consumer.” The Internet brought the real-place qualities of the ancient agora to modern business, requiring the destruction of what Weinberger called “Fort Business.” Surviving companies would be staffed by employees free to speak and participate with the market, across firewalls both real and virtual. Five years later, that’s the direction DIY-IT is taking us.

In early 1999, four computer industry veterans — Rick Levine, Christopher Locke, David Weinberger and I [Doc Searls] — nailed 95 theses up on the Web at the domain Cluetrain.com. Recalling Karl Marx as much as Martin Luther, we called the collection The Cluetrain Manifesto. The name came from an old Silicon Valley epitaph: The clue train stopped there four times a day for ten years and they never took delivery. A book followed in January 2000, just in time for the dot-com crash (a coincidence some found hard to ignore), and became a nonfiction bestseller.

Cluetrain's main beef was with the way markets were characterized, especially by dot-com companies that saw the Net as TV with crosshairs. As Locke put it, “We are not eyeballs or seats or end users or consumers. We are human beings and our reach exceeds your grasp. Deal with it.” Cluetrain saw the Net as an enabling condition that increased the power of demand at least as much as that of supply (the side then getting so much funding from VCs): “Through the Internet, people are discovering and inventing new ways to share relevant knowledge with blinding speed. As a direct result, markets are getting smarter — and getting smarter faster than most companies…”

Ofoto's DIY innovations include an offer-management system for merchandising and promotion, a customer service management system, a dynamic content-serving system, an active storage partition management tool, a server cache tool and a database framework that the company calls “a persistence layer.” Hertz says, “It’s easier to justify [the cost of developing internally] something that lets us offer new capabilities to our customers, or gives them better quality of service, or reduces our costs of doing business with our customers.” For example, he says, “In the case of our content management system, we saved money, came to market faster than we would have otherwise, and enabled our marketing people to do more with the service – to be more effective with our customers.”

Ofoto buys packaged products “if it’s for internal use and ‘good enough,” says Hertz. The biggest purchase was, and remains, Ofoto's original relational database management system (RDBMS), which was chosen “partly because it was easier to integrate into a custom solution that we were building and were designing internally.”
That RDBMS is perhaps the company’s only vendor solution that’s also close to its core IT competencies. Ofoto’s other turnkey applications are in areas where “we just want to put up some software on a box and let it go to work.” One example is Microsoft Exchange. “We tend to look at those kinds of things as overhead,” he says. “They’re part of the cost of operating.”

Among the semi-custom products Ofoto uses is an application server from a small vendor. Even here, Hertz says, there is a DIY motivation. “We are allergic to vertically integrated app[lication] servers that try to do all the work for you. But at a low level we had a specific need for better performance from our servlet engine. Rather than build our own, we found an app[lication] server did the job. The performance improvements resulted in dramatic capital equipment savings. So here we saved money by buying something, while in other cases we saved money by building it ourselves. It has to do with where we want to put our creative energy. It’s just a cost-benefit analysis issue.”
But it’s not always about saving money. “Sometimes it’s a matter of helping out operationally,” Hertz says. “For example, our [internally developed] partition manager made us so much more efficient operationally. We didn’t have any off-the-shelf alternative in that case.”

One key to running a DIY-IT shop is to have a talented and experienced staff. “You need to be able to meet problems head-on. If something comes up, you’re lucky if there’s an off-the-shelf product that does the job. In our business, there usually are no alternatives. DIY often is our only choice.” Another is to make the right technology choices to ensure a rich pool of talent is available. “The problems we’re solving are interesting challenges for our engineers. We have no trouble hiring people with the technical skills we need,” he says.

Ofoto is typical of other successful Web companies in that its IT personnel and infrastructures are essentially zero-based. Whatever legacies they have are not very old. In geological terms, they’re like Hawaii or Iceland: volcanic islands that are still growing and emerging from the sea. Their environment is boiling with activity. Says Hertz, “We’re in this milieu of people developing custom solutions for the Web. We’re not timid to roll up our sleeves and do exactly the thing that we need.” In contrast, companies whose businesses are not primarily conducted on the Web typically are structured and staffed to buy off-the-shelf solutions, to have something custom-developed for them, or to outsource completely. “The engineering services business is for people who can’t DIY,” he asserts.

The bottom line isn’t about technology. It’s about people. On that issue, Hertz quotes Steve Jobs, his old boss at Apple and NeXT: “It doesn’t make sense to hire smart people and then tell them what to do; we hired smart people so they could tell us what to do.”

**(Un)complicating Matters: Open-sourcing DIY-IT**

Open-source building material can come from the traditional communities of developers who do it out of passion and for recognition. But it can also come from companies that have developed something internally that they are willing to share.

There may be an inherent contradiction in open-sourcing the results of DIY projects – not because of greed, but for the reasons discovered by Lefkowitz during his
billing-system experiment: lack of motivation and lack of time. Who wants someone else’s customized work? And what motivates the creator to open it up?

But for smart vendors, open-sourcing some of one’s IP often makes sense. By doing so, they can share and use community improvements and sell some related product or service into a new market standard – one that they created.

One early example was when Bob Metcalfe, the primary inventor of Ethernet when he worked for Xerox PARC, persuaded Digital, Intel and Xerox to make Ethernet available as an open standard that anybody could use. It wasn’t exactly open-source (Xerox owned the patent), but it worked as a free standard. Metcalfe took advantage of that standard by launching 3Com, a new networking company that put Ethernet to use. Metcalfe rescued Ethernet from the laboratory and put it in the marketplace.

Apple’s mostly free-to-use “i” applications (iPhoto, iTunes, iMovie, iDVD and GarageBand) are a variant of that strategy. The company has done far more than place extremely desirable products into the hands of users. It has given users DIY tools that grow many markets at once: for its hardware (Macs, iMacs, iPods) as well as other PCs, for camcorders and digital cameras, for musical instruments – and, ultimately, for audio recordings and movies.

RealNetworks (page 19) open-sourced its Helix platform because one way or another it makes the things Real does sell more attractive – better QA’ed, more standardized – and because its products can benefit from shared value-added for which the users are bearing the costs. What it’s selling may be software platforms or data services or even hardware.

Open-sourcing internal applications can also help the problem of redundancy. DIY-IT professionals, since they work for someone, are willing to share their results not just out of altruism but also out of a kind of environmental concern for effective use of resources. They see themselves as part of a discipline more than as part of a company; they’re part of the mobile workforce that moves from company to company within a field. They don’t like to reinvent the wheel – not because it’s expensive, but because it is boring. Why not share the burden of inventing it – and of supporting and enhancing it thereafter? “The team who put OpenAdaptor together were working to solve the problem for the third time in different organizations,” says J.P. Rangaswami, CIO of DrKW. They understand – with Nicholas Carr – that IT alone is no longer a competitive advantage; it’s what you do with the IT that counts. That’s up to the business people who use the software.
DrKW’s OpenAdaptor: Open to improvement

It’s still relatively rare for corporate users, as opposed to vendors or software development companies, to open-source their own DIY projects. After all, they aren’t selling anything.

Or perhaps they are. In the case of investment bank Dresdner Kleinwort Wasserstein (DrKW) and its product OpenAdaptor, the “product” was jobs. Says Rangaswami, “Our initial reason was retention. By open-sourcing OpenAdaptor, we signaled to our staff that we were willing to embrace the brave new world of open source [and] that we were a cool place to work, way back in 1999. And it worked. We attracted and retained talent. Of 12 recent hires, seven chose DrKW because of our position on open source, being hosts and contributors rather than just consumers.” Adds Steve Howe, VP and global head of DrKW’s open source initiative. “It [DIY/open-sourcing] has done quite a lot for the reputation of DrKW. We’re punching above our weight: We’re pretty small, but now we have a good reputation. It has enabled us to attract some very high-caliber people.”

OpenAdaptor itself is an integration toolkit, a Java/XML-based Enterprise Application Integration platform which is particularly useful in a financial environment for passing messages between disparate systems for activities such as securities trading and risk management and for specific reporting and regulatory requirements. In a market where talent is scarce, the talent likes to see itself used to maximum effect – both on interesting projects rather than reinventing the wheel, and in a way that spreads the impact and reputation of their work. That argues for DIY work adding value to existing capabilities, and sharing that work across an audience broader than any single user. “The organization gains because scarce high-quality skills can be applied higher up the value chain while a larger community shares the [development] cost of commoditized layers” that are mass-produced and sold by vendors, says Rangaswami. He adds: “Software is becoming like the movie business. Very high production costs forcing people to change location and reuse stock shots. You can play limited-release, rental, shrink-wrap for purchase or free-to-air. [There’s a] high correlation of return when working with branded cast members. [There’s a] small number of genres: some regional, some global. [There’s the] concept of fast iteration for production, multiple takes before you wrap it.”

The decision to open-source OpenAdaptor, controversial within DrKW in 1999, has been a success. Says Rangaswami: “Thousands of downloads, hundreds of institutions... Only tens prepared to be quoted. [Yet other European banks] like Deutsche Bank and Abbey are prepared to endorse their usage publicly, [even] while compet-
ing with us.” Many other DIYers at those and other banks have contributed comments, bug fixes and the like. In addition, notes Howe, a company called Allianz Dresdner Asset Management in Germany wrote a new GUI for it. “It would have taken us about a year to write it ourselves,” says Howe. HP in the UK has also supported the project – which adds value to its computers – by using it for its own processing internally and by working on internationalization capabilities.

And, says Rangaswami, “It has been successful. We have cut our costs down to one-sixth of the original. There is an open source rule of thumb: for every 1000 people who observe, 80 are involved and 20 are truly active. This rule of thumb shows in the way people use and contribute to OpenAdaptor.”

The project continues, with version 3 due shortly. This time, says Howe, rather than put out a finished system as it did previously, DrKW will open up the beta version. “We’re trying to get the external community involved from the beginning,” he says. Rather than do all the work itself and benefit from improvements, DrKW hopes to get useful input from the start. “We’ve all got to make sure that it really works. We’re not just using it to order T-shirts. We’re managing 50-million Euro trades.”

Cost advantage...but not competitive advantage
Though DrKW’s costs and its competitors’ costs are down, you can reliably bet that there’s no strategic advantage in OpenAdaptor – or DrKW would not be sharing it.

OpenAdaptor is still a pretty generic tool. You could ask: What’s the difference between do-it-yourself application development and customization of existing applications? What’s the difference between a flexible application and a set of domain-specific tools?

Says Rangaswami: I think that open-source places the right perspective on the Nicholas Carr argument [about IT mattering]. It allows the industry to collaborate on utility aspects efficiently and elegantly, freeing up the talent to focus on newer challenges closer to the point of consumption: visualization tools, collaborative agents, genetic algorithms, predictive software in general, in a better architected and managed environment.”

As noted above, open-sourcing can relieve some redundancy by effectively outsourcing development. Then again, so can commercial software development. The challenge is to spread the fruits of DIY efforts broadly. Effective and efficient open-sourcing of DIY work is key in making the do-it-yourself world, with its flexi-
bility and independence, as economically efficient as the packaged-product world. As DIY becomes easier because of easier-to-use tools and more sophisticated users, the trade-offs are changing, but the ROI of a successful packaged product (as opposed to the many that languish in the space between development and broad deployment) can be huge.

In the end, DIY is about two things: building what you really need when it’s not there – where sharing costs and results makes sense – and building something you don’t want anyone else to have – where open source is anathema.

And indeed the do-it-yourselfers of the future – closer to ultimate customers and to business advantage – will be less likely to share than today’s techies. (At the user end, as we illustrated last month, small-business do-it-yourselfers are using tools such as Google, eBay and Yahoo! to run their businesses, but most of their DIY effort is redundant rather than re-used. See RELEASE 1.0, APRIL 2004.)

Right now, as the industry still doesn’t supply users’ needs effectively, open source and DIY are natural allies. But long run, we expect DIY to flourish in a closed way on top of shared, open-source tools.

**RealNetworks: Do-it-yourself market**

In the summer of 2002, RealNetworks announced Helix Community, a collaborative open-source development project for the Helix DNA platform, billed as “the first open multi-format platform for digital media creation, delivery and playback.” Helix is descended from Real’s former proprietary server platform. Of course, it’s still proprietary in its own way, but the code is open and available for enhancements or applications from third parties. It runs on all major operating systems and thus allows content providers to format content once for many delivery systems, including cell phones, and it allows vendors of those delivery systems to know that they are developing for a software platform that will attract lots of content.

The idea was to encourage do-it-yourselfers, thereby making the platform more valuable and ubiquitous, and providing an installed base into which Real can sell its content services. Kevin Foreman, general manager for Helix, says that the DNA community now includes 50,000 members. Of those, a few are vendors such as Nokia and Palm, who pay royalties and who represent the vast majority of the development efforts so far.
However, says Foreman, “We wanted to make it useful not just for NBC and giants like that, but also for CIOs developing corporate training videos and for people who want short clips, not necessarily full-length videos – just the Saddam capture, the weather, maybe a sports highlight.”

And it’s working, he says. “We’ve got lots of volunteer contributors. People start off as users: They give us feedback and bug fixes. Then they start to scratch their own itch, and develop something new, such as the MusicBrainz project.” That project uses an open-source registry of music CDs (akin to CDDB; see **Release 1.0, September 2003** to let users identify the CDs they have.

And Foreman expects many more: “A lot of universities and students are working on projects, but they haven’t rolled things out yet. Microsoft can out-engineer us, but they can’t out-engineer the world. We have more engineering resources applied than Microsoft does, even though we don’t pay them all.”

The Helix Community is also a high-profile project for Collabnet, a company founded by Brian Behlendorf (the prime mover behind Apache, the open-source server behind two-thirds of the Web’s pages). Collabnet’s field is CSD, or Collaborative Software Development. CSD is one evolutionary stage, or branch, of DIY-IT. Essentially it organizes and helps manage development projects that may be spread across multiple developers, teams, companies and countries. Collabnet’s SourceCast, for example, provides a way to manage multiple contributions to evolving source code, as well as knowledge management, communication management and many other capabilities. While Collabnet talks enterprise software project lingo, its roots are in the open source movement. Collabnet also helped DrKW with its open-source project. (see above, page 17.)

**DIY Future**

DIY is the phenomenon of people making their own software. It results from the happy collision of two trends – better-educated, more experience users, and easier-to-use, more transparent, more modular, more easily modifiable software. It’s also a cultural phenomenon – users who want to do it my way.
But it has economic impact – on everything from whether resources are allocated redundantly or leveraged, to the quality of software people use and the processes it supports (or doesn't). Over the long run, DIY means two things: First, that the value added shifts increasingly to the users from the vendors. And second, it should mean that users are better able to implement their own tasks in software.

DIY flourishes in the world of small projects – users building for themselves, rather than a virtual factory such as an ERP system. Thus it has little to do with one form of business competitive advantage – sheer efficiency of operations.

But it has everything to do with the other source of competitive advantage – enabling extraordinary, unique people to implement new, unique ideas, or simply supporting those people in doing their own tasks, their own way.

The middle of the road – where systems do routine things adequately – should get competed away. . .although markets move slowly and trends take a long time to play out. Users can benefit by hiring good people who can do it themselves effectively. And vendors need to figure out their proper place in this new universe: Either they can sell commodity systems, designed to do their work efficiently. Or they can sell tools and services that can help users develop new capabilities for themselves.
Resources & Contact Information

Nicholas Carr, ncarr@mac.com, www.nicholasgcarr.com
Christopher Locke, 1 (303) 413-9583; clocke@panix.com; www.rageboy.com
David Weinberger, self@evident.com; www.hyperorg.com
Robert M. Lefkowitz, AT&T Wireless, 1 (425) 288-7885; r0ml@attws.com
Euan Semple, BBC, 44 (208) 225-9704; euan.seemple@bbc.co.uk
JP Rangaswami, Steve Howe, Dresdner Kleinwort Wasserstein, steve.howe@drkw.com, jp@drkw.com
Wayne Rosing, Google, wayne@google.com
Don Marti, Linux Journal, ljeditor@ssc.com; www.linuxjournal.com
Marten Mickos, MySQL, 358 (40) 500-1153; marten@mysql.com
Adam Hertz, Ofoto, 1 (510) 229-1110; adan@ofoto.com
Craig Burton, PatchLink, 1 (480) 970-1025 x221; craig@craigburton.com
Kevin Foreman, RealNetworks, 1 (206) 674-2244; kevinf@real.com
Rick Levine, Wall Street On Demand, 1 (303) 583-4255; rick.levine@levinesquared.com

For further reading:
Nicholas Carr, “Does IT Matter? Information Technology and the Corrosion of Competitive Advantage,”
   http://www.nicholasgcarr.com/doesitmatter.html
Christopher Locke, Rick Levine, Doc Searls, David Weinberger, The Cluetrain Manifesto, (Perseus Books Group,
   February 2000), http://www.cluetrain.com

For further information:
Free Software Foundation and General Public License website: http://www.gnu.org
Helix Community: http://helixcommunity.org
## Calendar of High-Tech Events

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Location</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUNE 6-8</td>
<td><strong>D: All Things Digital</strong> - Carlsbad, CA. Hosted by Walt Mossberg and Kara Swisher, D explores the most pressing issues in the digital arena. This year's speakers include Bill Gates, Steve Jobs, Carla Fiorina, Larry Ellison, and a host of other key IT names. Register online or call 1 (866)416-3975 or 1 (650) 574-3975. d.wsj.com</td>
<td>Carlsbad, CA</td>
<td>Free Text</td>
</tr>
<tr>
<td>JUNE 15-18</td>
<td><strong>CommunicAsia 2004</strong> - Singapore. The biggest communications and IT trade show in Asia. For more information, visit the website or contact Yusnita Baharuddin, +65 6233-8621; fax, +65 6233-8715; e-mail, <a href="mailto:yus@sesallworld.com">yus@sesallworld.com</a>. <a href="http://www.CommunicAsia.com">www.CommunicAsia.com</a></td>
<td>Singapore</td>
<td>Free Text</td>
</tr>
<tr>
<td>JUNE 23-25</td>
<td><strong>AO2004</strong> - Palo Alto, CA. The Always-On community comes to life at AO2004, where global technology, government and media leaders will debate the opportunities and political implications surrounding always-on technologies. Register online or contact Kathy Osweiler, (415) 751-0170; e-mail, <a href="mailto:kathy@alwayson-network.com">kathy@alwayson-network.com</a>. <a href="http://www.alwayson-network.com/events">www.alwayson-network.com/events</a></td>
<td>Palo Alto, CA</td>
<td>Free Text</td>
</tr>
<tr>
<td>JUNE 24-25</td>
<td><strong>Supernova</strong> - Santa Clara, CA. In its third year, Supernova brings together 200 executives, technologists, investors, and other thought leaders to explore the decentralization of communications, software, and media. For more information, visit the website or e-mail <a href="mailto:info@supernova2004.com">info@supernova2004.com</a> <a href="http://www.supernova2004.com">www.supernova2004.com</a></td>
<td>Santa Clara, CA</td>
<td>Free Text</td>
</tr>
<tr>
<td>JULY 1-2</td>
<td><strong>Re-Thinking the Digital Future: Impacts, Trends and Challenges</strong> - Wolfsberg, Switzerland This event in the Wolfsberg Think Tank series examines the Internet and its implications on society as well as its impact on learning and business. Register online. For more information contact Professor Prabhu Guptara, <a href="mailto:prabhu.guptara@ubs.com">prabhu.guptara@ubs.com</a>. <a href="http://www.wolfsberg.com">www.wolfsberg.com</a></td>
<td>Wolfsberg, Switzerland</td>
<td>Free Text</td>
</tr>
<tr>
<td>JULY 1-2</td>
<td><strong>O'Reilly Open Source Convention</strong> - Portland, OR. This year's theme is &quot;Opening the Future: Discover, Develop, Deliver.&quot; Open source leaders gather to push the boundaries of their respective technologies. Featuring our own Esther Dyson. Register online or contact Linda Holder, <a href="mailto:lholder@oreilly.com">lholder@oreilly.com</a>. conferences.oreillynet.com/os2004/</td>
<td>Portland, OR</td>
<td>Free Text</td>
</tr>
<tr>
<td>SEPTEMBER 8-10</td>
<td><strong>Demo Mobile</strong> - La Jolla, CA. See the products, take in the speeches and hear the interviews from the people and companies that are driving the mobile marketplace. Register online or call IDG at 1 (800) 643-4668 or 1 (508) 490-6545; fax, 1 (508) 460-1385; <a href="mailto:registrar@idgexecforums.com">registrar@idgexecforums.com</a>. <a href="http://www.demo.com">www.demo.com</a></td>
<td>La Jolla, CA</td>
<td>Free Text</td>
</tr>
<tr>
<td>OCTOBER 12-15</td>
<td><strong>TedMed3</strong> - Charleston, SC. Discover how technology can help you achieve a healthier life. Imagine! For more information contact Richard Saul Wurman, (401) 848-2299; e-mail, <a href="mailto:wurmanrs@aol.com">wurmanrs@aol.com</a>. <a href="http://www.tedmed.com">www.tedmed.com</a></td>
<td>Charleston, SC</td>
<td>Free Text</td>
</tr>
</tbody>
</table>

**Footnotes:**
- Events Esther plans to attend.

Lack of a symbol is no indication of lack of merit. The full, current calendar is available on our website, www.release1-0.com. Please contact Christina Koukkos (christina@edventure.com) to let us know about other events we should include.
Get our RSS feeds: We now have RSS feeds for Release 1.0 and Esther’s weblog.
Subscribe to them at http://www.edventure.com/rss

Release 1.0 Subscription Form

Complete this form and join the other industry executives who regularly rely on Release 1.0 to stay ahead of the headlines. Or if you wish, you can also subscribe online at www.release1-0.com.

Your annual Release 1.0 subscription costs $795 per year ($850 outside the US, Canada and Mexico), and includes both the print and electronic versions of 11 monthly issues; 25% off the cover price when you order from our online archives; a Release 1.0 binder; the bound transcript of this year’s PC Forum (a $300 value) and an invitation to next year’s PC Forum.

NAME _____________________________________________________________

TITLE ____________________________ COMPANY ___________________________

ADDRESS ___________________________________________________________

CITY _____________________________ STATE _______ ZIP __________ COUNTRY ____________

TELEPHONE ___________________________ FAX ___________________________

E-MAIL* _______________________________ URL ___________________________

*personal e-mail address required for electronic access.

☐ My colleagues should read Release 1.0, too!
Send me information about multiple copy subscriptions and electronic site licenses.

☐ Check enclosed ☐ Charge my (circle one): AMERICAN EXPRESS MASTER CARD VISA

CARD NUMBER ___________________________ EXPIRATION DATE ______________

NAME AND BILLING ADDRESS _____________________________________________

SIGNATURE ___________________________________________________________

Please fax this form to Brodie Crawford at 1 (212) 924-0240.

Payment must be included with this form. Your satisfaction is guaranteed or your money back.

If you wish to pay by check, please mail this form with payment to: EDventure Holdings, 104 Fifth Avenue, 20th Floor, New York, NY 10011, USA. If you have any questions, please call us at 1 (212) 924-8800; e-mail us@edventure.com; www.release1-0.com.