Extraordinary group efforts don't have to be miraculous or accidental. An environment designed to produce cheap, plentiful transactions unleashes collaborations that break through organizational barriers.

# Collaboration Rules

by Philip Evans and Bob Wolf

**Corporate leaders seeking growth,** learning, and innovation may find the answer in a surprising place: the open-source software community. Unknowingly, perhaps, the folks who brought you Linux are virtuoso practitioners of new work principles that produce energized teams and lower costs. Nor are they alone.

By any measure, Linux is a powerfully competitive product. It is estimated that more servers run on Linux than on any other operating system. It has overwhelmed UNIX as a commercial offering. And its advantages extend beyond cost and quality to the speed with which it is enhanced and improved. While partisans debate its technical limitations and treatment of intellectual property, they agree that the product's success is inseparable from its distinctive mode of production. Specifically, Linux is the creation of an essentially voluntary, self-organizing community of thousands of programmers and companies. Most leaders would sell their grandmothers for workforces that collaborate as efficiently, frictionlessly, and creatively as the self-styled Linux "hackers."

But Linux is software, and software is kind of weird. Toyota, however, is a company like any other–any other consistently ranked among the world's top-performing organizations, that is. The auto-

PHILIPPE WEISBECKER

maker has long been a leader in quality and lean production, and the success of the hybrid Prius has established its reputation as an innovator. We have found that Toyota's managerial methods resemble, in a number of their fundamentals, the workings of the Linux community; the Toyota Production System (TPS) owes some of its vaunted responsiveness to open-source traits. In fact, Toyota itself is evolving into a hybrid between a conventional hierarchy and a Linux-like self-organizing network.

(Throughout this article, we use the term "Linux" as shorthand for the free/open-source software community that developed and continues to refine the operating system and other open-source programs. We use "Toyota" as shorthand for the Toyota Production System, which comprises Toyota and its direct–"tier one" in automotive parlance–suppliers in Japan and the United States.)

Toyota is remarkably similar to Linux in the way it blends key characteristics of both markets and hierarchies. Like markets, the Toyota and Linux communities can be self-organizing, but unlike markets, they don't use cash or contracts at critical junctures. Like hierarchies, Toyota and Linux enjoy low transaction costs, but unlike hierarchies, their members may belong to many different organizations (or to none at all) and are not corseted by specific, predefined roles and responsibilities. And like hierarchies, members share a common purpose, but that purpose emanates from self-motivation rather than from the external incentives or sanctions that hierarchies generally impose. In these respects, Toyota and Linux represent the best of both worlds. An analysis of their common characteristics suggests how high-performance organizations remain productive and inventive even under grueling conditions. We believe those lessons can significantly improve the way work in most organizations gets done.

**Few communities appear more different than the anarchistic, caffeinated, hirsute world of hackers and the disciplined, tea-sipping, clean-cut world of Japanese auto engineering.**

## Tuesday, December 2, 2003

Near midnight, Andrea Barisani, system administrator in the physics department of the University of Trieste, discovered that an attacker had struck his institution's Gentoo Linux server. He traced the breach to a vulnerable spot in the Linux kernel and another in rsync, a file transfer mechanism that automatically replicates data among computers. This was a serious attack: Any penetration of rsync could compromise files in thousands of servers worldwide.

Barisani woke some colleagues, who put him in touch with Mike Warfield, a senior researcher at Internet Security Systems in Atlanta, and with Andrew "Tridge" Tridgell, a well-known Linux programmer in Australia on whose doctoral thesis rsync was based. They directed Barisani's message (made anonymous for security reasons) to another Australian, Martin Pool, who worked for Hewlett-Packard in Canberra and had been a leader in rsync's development. Although Pool was no longer responsible for rsync (nobody was), he immediately hit the phones and e-mail, first quizzing Warfield and Dave Dykstra (another early contributor to rsync's development, who was based in California) about vulnerabilities and then helping Barisani trace the failure line by line.

By morning Trieste time, Pool and Barisani had found the precise location of the breach. Pool contacted the current rsync development group, while Barisani connected with the loose affiliation of amateurs and professionals that package Gentoo Linux, and he posted an early warning advisory to the Gentoo site. Pool and Paul "Rusty" Russell (a fellow Canberran who works for IBM) then labored through the Australian night to write a patch, and within five hours Gentoo user-developers started testing the first version. Meanwhile, Tridge crafted a description of the vulnerability and its fix, being sure (at Pool's urging) to credit Barisani and Warfield for their behind-the-scenes efforts. On Thursday afternoon Canberra time, the announcement and the patch were posted to the rsync Web site and thus distributed to Linux users worldwide.

A few days after the emergency, having caught up on his sleep, Barisani volunteered to collaborate with Warfield in setting up a system of deliberately vulnerable servers to lure the system cracker into revealing himself.

***Philip Evans*** *(evans.philip@bcg.com) is a senior vice president, and* ***Bob Wolf*** *(wolf.bob@bcg.com) is a manager, in the Boston office of the Boston Consulting Group. Evans is a coauthor of* Blown to Bits: How the New Economics of Information Transforms Strategy *(Harvard Business School Press, 1999).*

No one authorized or directed this effort. No one – amateur or professional – was paid for participating or would have been sanctioned for not doing so. No one's job hinged on stopping the attack. No one clammed up for fear of legal liability. Indeed, the larger user community was kept informed of all developments. Yet despite the need for the highest security, a group of some 20 people, scarcely any of whom had ever met, employed by a dozen different companies, living in as many time zones and straying far from their job descriptions, accomplished in about 29 hours what might have taken colleagues in adjacent cubicles weeks or months.

It's tempting to dismiss this as an example of hacker weirdness – admirable, yes, but nothing to do with real business. Consider, however, another story.

## Saturday, February 1, 1997

At 4:18 AM, a fire broke out in the Kariya Number 1 plant of Aisin Seiki, a major Japanese automotive parts supplier. Within minutes, the building and virtually all the specialized machinery inside were destroyed. Kariya Number 1 produces 99% of the brake fluid–proportioning valves, or P-valves, for Toyota's Japanese operations – parts required by every vehicle Toyota builds. And Toyota, true to its just-in-time principles, had less than a day's inventory. The Japanese Toyota Production System faced the possibility of a total shutdown lasting months.

Within hours, Aisin engineers met with their counterparts at Toyota and Toyota's other tier one suppliers. The group agreed to improvise as much production as possible. As news spread through the supplier network, some tier twos volunteered to play leadership roles. Aisin sent blueprints for the valves to any supplier that requested them and distributed whatever undamaged tools, raw materials, and work in process could be salvaged. Aisin and Toyota engineers helped jury-rig production lines in 62 locations – unused machine shops, Toyota's own prototyping shop, even a sewing machine facility owned by Brother. Denso, Toyota's largest supplier, volunteered to manage the messy logistics of shipping valves to Aisin for inspection and then on to Toyota's stalled assembly lines.

Everyone was surprised when a small tier two supplier of welding electrodes, Kyoritsu Sangyo, was first to deliver production-quality valves to Toyota – 1,000 of them, just 85 hours after the fire. Others followed rapidly, and Toyota started reopening assembly lines on Wednesday. Roughly two weeks after the halt, the entire supply chain was back to full production. Six months later, Aisin distributed an emergency response guide containing lessons drawn from the experience and recommending procedures for responding to such situations in the future.

No one individual or organization planned this effort: rather, people and companies stepped in where they could. Competitors collaborated. No one at the time was paid for contributing. Months later, Aisin compensated the other companies for the direct costs of the valves they had delivered. Toyota gave each tier one supplier an honorarium based on current sales to the automaker, encouraging – but not requiring – them to do likewise for their own tier twos.

Few communities appear more different than the anarchistic, caffeinated, hirsute world of hackers and the disciplined, tea-sipping, clean-cut world of Japanese auto

## Building Vibrant Human Networks

Companies laying the groundwork for high-performance collaboration should follow these principles:

**Deploy pervasive collaborative technology.** Keep it simple and open: "small pieces loosely joined," in *Cluetrain Manifesto* coauthor David Weinberger's felicitous phrase. Tools should work together through common standards and be as compatible as possible with those of the rest of the world. Think options not integration, adaptability not static efficiency.

**Keep work visible.** Unless there is a really good reason not to, let everybody see everybody's real work. Let people learn to filter and sort for themselves. Don't abstract, summarize, or channel. Fodder is good. Put it within everyone's reach.

**Build communities of trust.** When people trust one another, they are more likely to collaborate freely and productively. When people trust their organizations, they are more likely to give of themselves now in anticipation of future reward. And when organizations trust each other, they are more likely to share intellectual property without choking on legalisms.

**Think modularly.** Reengineering was about thinking linearly: managing the end-to-end process instead of discrete functions. That approach fosters focused efficiency but inhibits variety and adaptability. Modularity is the reverse: sacrificing static efficiency for the recombinant value of options. Think modular teams as well as processes. The finer, the better.

**Encourage teaming.** Celebrate the sacrifices that teams make for the broader enterprise, including customers and suppliers. Dismantle individualized performance metrics and rewards that pit people against one another. Cheap transactions among the many fuel more innovation than expensive incentives aimed at the few. Reward the group, and the group will reward you.

engineering. But the parallels between these stories are striking. In both of them, individuals found one another and stepped into roles without a plan or an established command-and-control structure. An extended human network organized itself in hours and "swarmed" against a threat. People, teams, and companies worked together without legal contracts or negotiated payment. And despite the lack of any authoritarian stick or financial carrot, those people worked *like hell* to solve the problem.

Now, obviously, these were emergency responses. But a look at the day-to-day operations of the Linux community and the Toyota Production System reveals that those responses were merely intensifications of the way people were already working.

## Obsession, Interaction, and a Light Touch

The rules of markets are about cash and contracts. The rules of hierarchies are about authority and accountability. But at the core of the Linux and Toyota communities are rules about three entirely different things: how individuals and small groups work together; how, and how widely, they communicate; and how leaders guide them toward a common goal.

**A Common Work Discipline.** The Linux and Toyota communities are both composed of engineers, so members have the same skills as their colleagues and speak the same language. But these groups are far more disciplined and rigorous in their approach to work than are other engineering communities. Both emphasize granularity: They pay attention to small details, eliminate problems at the source, and trim anything resembling excess, whether it be work, code, or material. Linux members, for example, share an obsession with writing minimal code, compiling each day's output before proceeding to the next and extirpating programming flaws as they go along. For their part, TPS engineers are relentless in applying short cycles of trial and error, focusing on just one thing at a time, and getting inside and observing actual processes. Both groups carry those principles to apparent extremes. Linux programmers whittle away at code in pursuit not of computational efficiency but of elegance. Toyota engineers reject stampings for the Lexus hood – while flawless and entirely within spec – because the sheen, to their eyes, lacks luster.

**Widespread, Granular Communication.** In both the Linux and Toyota communities, information about problems and solutions is shared widely, frequently, and in small increments. Most Linux hacker communication is not between individuals but by postings to open, searchable Listservs. Anyone can review the version history of the code and the Listserv debates – not executive summaries or abstracts but the raw activity itself. And every code contribution is stress tested by scores of people. As a famous open-source mixed metaphor puts it: "With a thousand eyes, all bugs are shallow." The median upload to the Linux kernel is a mere dozen lines of code. The working alpha version is recompiled every 24 hours, so hackers reconcile their efforts almost continuously. If someone worked in isolation for six months on even the most brilliant contribution, it would probably be rejected for lack of compatibility with the others' efforts.

The Toyota philosophy of continuous improvement likewise comprises a thousand small collaborations. Toyota engineers are famously drilled to "ask why five times" to follow a chain of causes and effects back to a problem's root. This is not a vapid cliché about thinking deeply. Quite the contrary, in fact. The precept's merit is precisely in its superficiality. Saying that B causes A is simplistic – all the complexities of multiple interactions boiled down to a single cause and effect. But the chain of thought required to discover that C causes B, and D causes C, quickly takes you into a new domain, probably someone else's. So rather than concoct complex solutions within their own domains, engineers must seek simple ones beyond them. "Doing your why-whys," as the practice is known, is not about depth at all – it's about breadth.

And as with Linux, Toyota's communication protocols enforce this discipline. Each meeting addresses just one topic and drives toward a specific outcome, even if that means the same people meet more than once in a day. Lessons are written in a standard format on a single sheet of A3 paper. And everyone learns how to craft these reports, down to the fold in the document that shows the main points and conceals the details.

**Leaders as Connectors.** At every level, Linux and TPS leaders play three critical roles. They instruct community members–often by example–in the disciplines we've just described. They articulate clear and simple goals for each project based on their strategic vision. And they connect people, by merit of being very well connected themselves. The top Linux programmers process upwards of 300 or 400 e-mails daily. Fujio Cho, the president of Toyota, manages by similarly numerous daily interactions that transcend the normal chain of command.

Neither community treats leading as a discipline distinct from doing. Rather, the credibility and, therefore, authority of leaders derives from their proficiency as practitioners. The content of leaders' staccato communications is less *about* work than it *is* work. (When Linux creator Linus Torvalds dashes off his scores of daily e-mails, he writes almost as much in the C programming language as he does in English.)

Occasionally, leaders do have to perform traditional leadership acts, such as arbitrating conflicts. That, however, is the exception and is viewed as a bit of a system failure. The default assumption is that, as far as possible, managers don't manage in a traditional sense: The human network manages itself. In Linux, development priorities are decided not by a CEO but by thousands of hackers voting with their feet by choosing what to work on. That kind of radical self-management does not happen at Toyota, except in emergencies. But even in daily operations, a single production worker who sees a quality problem can stop the line, and project teams have wide latitude to tap resources, make purchase decisions, and pursue priorities they set for themselves.

Taken together, these three principles seed a continuously adapting system. Over and over, ideas are formulated in tight, testable packets; they are communicated with minimal attenuation through established, direct, person-to-person connections; and where links are absent, widely connected leader-practitioners create them as needed. This is discipline, but not the discipline of conformity produced by controls and incentives. Rather, it resembles the discipline of science. Like scientific communities, these systems rely on common procedures, common rules for communication and testing, and common goals clearly understood. Individual behavior is rigorously cautious, but collective achievement is marked by continuous, radical innovation.

**In the Linux and Toyota communities, leading is not treated as a discipline distinct from doing. Rather, the authority of leaders derives from their proficiency as practitioners.**

## What They Know and How They Know It

At the heart of Linux and the Toyota Production System, then, is a set of work, communication, and leadership practices that contributes to a new form of collaboration. This collaboration also relies on two infrastructure components: a shared pool of knowledge and universally available tools for moving knowledge around.

**Common Intellectual Property.** The General Public License under which Linux is published requires that all distributors make their source code freely available so that others can freely emend it. This viral principle prevents code from being stowed away in proprietary products. That transparency, in turn, breaks down the distinction between producer and user. A sophisticated "customer" like Andrea Barisani is really a user-developer, who fixes flaws and adds features for his own benefit, then shares those improvements with everyone else. Such a role is impossible when proprietary code is licensed from a commercial vendor. Similarly, Toyota's supply chain is predicated on the principle that while product knowledge (such as a blueprint) is someone's intellectual property, process knowledge is shared. That breaks down some distinctions among companies. Toyota's suppliers regularly share extensive process improvement lessons both vertically and laterally, even with their competitors. In Japan, suppliers are generally exclusive to a single OEM, so the collective benefit of that shared information stays within the Toyota supply chain. But even in the United States, where Toyota is just one of several customers for most of its tier ones, the carmaker does the same thing through its Bluegrass Automotive Manufacturers Association, which disseminates best practices to all members.

**Simple, Pervasive Technology.** Although information is the lifeblood of the Linux and TPS communities, their circulation systems are surprisingly rudimentary. Linux developers produce state-of-the-art software using communication technology no more sophisticated than e-mail and Listservs – but those mundane tools are used by everyone. Indeed, so great is the value placed on universality that plain-text, rather than formatted, e-mails are the norm, ensuring that messages will appear exactly the same to all recipients. Toyota, whose products are state-of-the-art as well, also prefers simple and pervasive internal technology. An empty kanban bin signals the need for parts replenishment; a length of duct tape on the assembly-line floor allots the completion times of tasks on a moving vehicle. Quality control problems on the assembly line are announced via pagers and TV monitors. And everyone gets the alert. Even Ray Tanguay, head of Toyota Canada, is paged whenever a flaw is found in the latest Lexus consignment on the dock in Long Beach, California, or in a service bay anywhere in North America.

**Monetary carrots and accountability sticks motivate people to perform narrow, specified tasks. Admiration and applause are far more effective stimulants of above-and-beyond behavior.**

## The Power of Trust and Applause

Such extremely rich, flexible collaborations have positive psychological consequences for participants and powerful competitive ones for their organizations. Those consequences are rich common knowledge, the ability to organize teams modularly, extraordinary motivation, and high levels of trust.

**Rich Semantic Knowledge.** A rigorous work discipline, common intellectual property, and constant sharing combine to distribute knowledge widely and relatively evenly across human networks. That knowledge includes not just the formal, syntactic information found in databases but also the semantically rich, ambiguous knowledge about content and process that is the currency of creative collaboration. What do we mean by the sheen of a body stamping having insufficient luster? What, precisely, must we discuss with the steel company to correct such an ill-defined problem? This kind of no-easy-answer question is continually discussed and resolved in a thousand small-team collaborations. The resulting nuanced thinking and richer common vocabulary on such matters are fed back into the knowledge pool, where they are available for further refinement by the whole community.

**Modular Teaming.** Modularity is a design principle by which a complex process or product is divided into simple parts connected by standard rules. In modular arrangements of teams, each team focuses on small, simple tasks that together make up a larger whole. Modularity allows an organization to run multiple, parallel experiments, making many small bets instead of a few large ones. The Toyota suppliers organized themselves this way to make P-valves, operating partly by direction but chiefly by volunteering to do what each knew best. The Gentoo group, Tridge's security experts, and Pool's circle of rsync alumni were preexisting and overlapping modules that mixed and matched roles as the emergency required.

When we mapped the patterns of day-to-day collaboration across the entire Linux kernel development effort, we found that such modular arrangements are pervasive and, to a degree, nest within one another. This creates a kind of dynamic organization chart – a chart that nobody wrote but one that enables the community to expand and adapt without collapsing into chaos.

**Intrinsic Motivation.** The Linux and TPS communities dissociate money from key transactions. Yet despite weak financial incentives, they command a level of motivation higher than that found in conventional environments. Monetary carrots and accountability sticks, psychologists have consistently found, motivate people to perform narrow, specified tasks but generally discourage people from going beyond them. Admiration and applause are far more effective stimulants of above-and-beyond behavior. "The personal reputation of the developer is attached to every release," Linus Torvalds explained to technology columnist Robert Cringely in 1998. "If you are making something to give away to the world, something that represents to millions of users your philosophy of computing, you will always make it the very best product you can."

Psychologists also emphasize the motivational importance of autonomy. Linux programmers decide for themselves how and where to contribute, and they enjoy the satisfaction of producing something whose quality is defined not by a marketing department nor by accountants but by their own exacting standards. Coauthor Bob Wolf and MIT's Karim Lakhani surveyed more than 800 user-developers, and over half said that their open-source work is the most valuable and creative endeavor in their professional lives.

The Toyota Production System doesn't offer such extreme autonomy, of course, and employees don't work for free. But compared with their counterparts in the rest of the auto industry, TPS workers enjoy fewer controls, greater encouragement of individual initiative, fewer metrics attached to individual performance, and louder peer applause. Professional and corporate pride, not Toyota's honorarium, was the payoff for the team at Kyoritsu Sangyo when it delivered the first batch of P-valves. That same pride is felt by a junior assembly-line worker when he is trusted by his peers to experiment with process improvements and to stop the line if something goes wrong.

**High Levels of Trust.** When information flows freely, reputation, more than reciprocity, becomes the basis for trust. Operating under constant scrutiny – which is challenging but not hostile – workers know their reputations are at risk, and that serves as a guarantor of good behavior, the equivalent of contracts in a market or audits in a hierarchy. Hence the obsession in the Linux community with acknowledging code contributions and including personal e-mail addresses in the comment fields of Listservs. Hence the generous public credit bestowed on Barisani and Warfield. Hence the collective celebration of Kyoritsu Sangyo's heroic efforts.

With their reputations at stake, people are less likely to act opportunistically. With the same information available to everyone, there is less chance that one party will exploit another's ignorance. And with a common vocabulary and way of working, fewer misunderstandings occur. Those factors drive up trust, the fundamental social capital of these communities.

Trust would matter less if there were no cost to exiting these networks or if transactions were of radically different sizes (since that would tempt people or companies to break the rules when a big opportunity arose). But in both the Linux and Toyota communities, entry to the inner circle is a hard-earned privilege, and both operate on many small exchanges.

And, of course, where trust is the currency, reputation is a source of power. In a sparse network, such as most markets and hierarchies, power derives from controlling or brokering the flow of information and often, therefore, from restricting it. In a dense network, however, information simply flows around the would-be choke point. Under those circumstances, there is more power in being an information source than an information sink. Consequently, individuals are motivated to maximize both the visibility of their work and their connections to those who are themselves broadly connected. That, in turn, feeds the information density of the network.

## Cheap Transactions and Plenty of Them

So far we have been discussing the content of work. But the TPS and Linux models change the economics of work as well, by driving down transaction costs. Low transaction costs make it profitable for organizations to perform more and smaller transactions – both internal and external – and so increase the pace and flexibility typical of high-performance organizations.

The classical sources of transaction costs are mutual vulnerability in the face of uncertainty, conflicting interests, and unequal access to information. We spend cash on negotiation, supervision, and restitution to reduce those imperfections. Both markets and hierarchies incur transaction costs (though hierarchies exist to economize on them, as Ronald Coase and Oliver Williamson have argued). Using a methodology developed by J.J. Wallis and Douglass North, we estimate that in the year 2000, cash transaction costs alone accounted for over half the nongovernmental U.S. GDP! We spend more money negotiating and enforcing transactions than we do fulfilling them.

In the Linux and Toyota communities, agreements are enforced not by the sanction of a legal contract, nor by the authority of a boss, but by mutual trust – lowering transaction costs dramatically. This is not new: Teams of people everywhere in the conventional workplace operate on the basis of trust.

What is new is how widely trust can extend, even to people who don't know each other – or even among those who have competing interests. Aisin trusted its rival suppliers with the P-valve blueprints. The rsync hackers swapped sensitive information with people they had never met.

### Giving Credit Where Credit Is Due

The Linux community uses a particular format – a "credit file" – to acknowledge the contributions of its members. If we, for instance, were to acknowledge in the Linux format the contributions of individuals who helped shape our thinking for this article, here's how it would look:

```
n: Mark Blaxill
e: blaxill.mark@bcg.com
d: Exploration of economics
   of open source
s: Boston Consulting Group

n: Paul Carlile
e: carlile@bu.edu
d: Discussion of Linux/
   Toyota parallels
s: Boston University

n: Karim Lakhani
e: lakhani@mit.edu
d: Discussion of Linux/Toyota
   parallels
d: Survey of free/open source
   hackers
s: MIT
```

Toyota's component suppliers share process knowledge daily, trusting that Toyota will not use it to beat down prices. Linux hackers trust one another to make uncoordinated and simultaneous emendations in the code base.

Moreover, holding property in common – as certain kinds of intellectual property are held within these communities – lowers the monetary stakes among the joint owners. Transaction costs fall because there is simply less to negotiate over. In the Linux community, transaction costs approach zero. Hewlett-Packard paid Martin Pool to be a Linux engineer, but it does not follow that HP needed to be paid on the margin for Pool's nocturnal labors on rsync. In the Toyota community, transaction costs, while not zero, have been radically reduced. When the Aisin Seiki plant was destroyed, Toyota and its suppliers didn't sue one another or cobble together emergency supply contracts. They simply got on with the job, trusting that fair restitution would eventually be made. Jeffrey Dyer, a professor of strategy at Brigham Young University, estimates that transaction costs between Toyota and its tier one suppliers are just one-eighth those at General Motors, a disparity he attributes to different levels of trust.

## A Model for Many

Bring together all these elements and you have a virtuous circle. A dense, self-organizing network creates the conditions for large-scale trust. Large-scale trust drives down transaction costs. Low transaction costs, in turn, enable lots of small transactions, which create a cumulatively deepening, self-organized network.

Once the system achieves critical mass, it feeds on itself. The larger the system, the more broadly shared the knowledge, language, and work style. The greater individuals' reputational capital, the louder the applause and the stronger the motivation. The success of Linux is evidence of the power of that virtuous circle. Toyota's success is evidence that it is also powerful in conventional, profit-maximizing companies.

The Linux community and Toyota Production System are strikingly different. The fact that they achieve so much in such similar ways points to some principles others can follow.

- The discipline of science is surprisingly adaptable to the organization of corporate – and even intercorporate – work.
- Under some circumstances, trust is a viable substitute for market contracts and hierarchical authority, not just in small teams but also in very large communities.
- Across supply chains, organizations that are able to substitute trust for contracts gain more from the collaboration than they lose in bargaining power.
- Low transaction costs buy more innovation than do high monetary incentives.

These principles serve businesses' need for growth and innovation in ways that traditional organizational models do not. And perhaps the effectiveness of these collaborations suggests the ultimate emergence of something altogether new. Not markets. Not hierarchies. But a powerful combination of both – and a signature of the networked society.

Reprint R0507H
To order, see page 195.

## Exploiting the Neglected 80%

The Pareto Principle famously dictates that companies derive 80% of their value from just 20% of their products, customers, or ideas. Because of high transaction costs, the long tail of that curve – that 80% of uncertain value generators – cannot be explored. So in the name of company focus, the tail gets lopped off, segmented away, or reengineered out of existence. Potentially profitable innovations die with it.

Organizations that reduce transaction costs can embrace the rejected 80%. They can respond to weak market signals, tap small segments, and experiment with unlikely combinations of technologies. They can place a hundred small bets instead of a few big ones.

For example, Detroit considered hybrid vehicles to be an uninteresting intermediate product: U.S. auto executives preferred so-far-unfulfilled research on fuel cell technology. Meanwhile, Toyota was building the Prius. The hybrid is now in its second generation, and Toyota expects to sell 300,000 worldwide this year. Toyota's low transaction costs and penchant for small-scale collaborations helped it keep open 80 discrete options for the hybrid engine until just six months before delivering a final design. Conventional automakers would have needed to freeze those design variables at least two years earlier.

It is in the interstices of the human network – rather than in the minds of a few wunderkinder – that most real innovations are born. And so it is transaction costs that constrain innovation by constraining opportunities to share different and conflicting ideas, skills, and prejudices.

"Detroit people are far more talented than people at Toyota," remarks Toyota president Fujio Cho, with excessive modesty. "But we take averagely talented people and make them work as spectacular teams." The network, in other words, is the innovator.