

# Open for Business

Open Source Inspired  
Innovation





# Open for Business

Jaap Bloem | Menno van Doorn | Erik van Ommeren

Open Source Inspired Innovation

2007

VINT | Vision • Inspiration • Navigation • Trends  
Research Institute for the Analysis of New Technology  
Sogeti Group



This work is licensed under the Creative Commons Attribution-No  
Derivative Works 3.0 Unported License. To view a copy of this li-  
cense, visit <http://creativecommons.org/licenses/by-nd/3.0/> or send  
a letter to Creative Commons,  
543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

We would like to thank the following for permission to use visual  
material: Brewtopia, BzzAgents, Cambrian House, ChicagoCrime.  
org, John Fluevog, *Fast Company*, Lego, Loughborough University,  
Nick Lowndes (Eastwing), Sun Microsystems and Threadless.

© 2007 Research Institute for the Analysis of New Technology VINT

*translation* Robert Olsen  
*copy editing* George Hall and Susan MacFarlane  
*production* LINE UP boek en media bv, Groningen (the Netherlands)  
*book design* Jan Faber  
*printing* Bariet, Ruinen (the Netherlands)  
*binding* Abbringh, Groningen (the Netherlands)

ISBN 978-90-75414-19-6

### The Rise of Crowdsourcing

"Now the productive potential of millions of plugged-in enthusiasts is attracting the attention of old-line businesses, too. For the last decade or so, companies have been looking overseas, to India or China, for cheap labor. But now it doesn't matter where the laborers are – they might be down the block, they might be in Indonesia – as long as they are connected to the network."

Jeff Howe in *Wired Magazine*, June 2006

"Just as it is now impossible to think about getting things done without at least considering the role that could be played by the internet, so will it soon be impossible to think about how to solve a large social problem without at least considering the role of methods originally and unintentionally pioneered by volunteer programmers just trying to build a better program."

Demos, think tank for everyday democracy, *Wide Open, Open Source Methods and Their Future Potential*

"The disruptive effects of the open source production process could be as great or greater outside the information sector, at first simply because of the increased efficiency of information processing that will effect many other economic activities, and second because of the spread of the model of production itself to other sectors of the economy."

Steven Weber, professor at the University of California, Berkeley and director of the Institute of International Studies



# Contents

<b>Foreword</b>	<b>9</b>
<b>1 The relationship between open source and open innovation</b>	<b>13</b>
1.1 Open source has many faces	15
1.2 Seven characteristics of open source	18
1.3 Open source inspiration for innovation	22
1.4 Crowdsourcing	27
1.5 The outline and structure of this book	34
<b>2 Open innovation – the new trend</b>	<b>41</b>
2.1 Innovation writ large	43
2.2 Value creation with open innovation	46
2.3 Have others do the work	48
2.4 Product innovation in the age of coding	50
2.5 Open innovation is something typical of our generation	57
2.6 Webservices: light-weight software for business innovation	61
2.7 Open innovation at Amazon, Google and eBay	63
2.8 Open innovation at Toyota	67
2.9 Open innovation at Procter & Gamble	70
2.10 Open innovation in seeker-solver networks	71
2.11 Open innovation in the press	73
2.12 Open innovation in advertising and marketing	74
2.13 Open source beer from Brewtopia	77
2.14 Open innovation at Boeing	80
2.15 Open innovation in building new computers	80
2.16 Open innovation à la CrowdSpirit	82
2.17 Open innovation at Wikipedia	84
2.18 Conclusions	85
<b>3 The software market has taken the lead</b>	<b>95</b>
3.1 Open source has transformed the software portfolio	97
3.2 The business model of the original open source organizations	100
3.3 The established order and the new wave get together	105
3.4 Microsoft: the favorite target of the “open source movement”	106
3.5 Oracle does not believe in open source business but still buys the companies	111
3.6 IBM is an open source evangelist with enormous closed source interests	114
3.7 HP specializes in hardware and services, and seizes open source opportunities	116

3.8 Sun produces its own hardware and software platforms, competes with Linux, and is pro-open source	117
3.9 Conclusions on open innovation	121
<b>4 What motivates an open source community?</b>	<b>127</b>
4.1 Open source software does not and can never amount to much	128
4.2 Hacking open source and the passion of the individual	130
4.3 Who are these open source developers?	132
4.4 Is there a taboo against making money?	134
4.5 Open source developers want to belong to the community	136
4.6 In this gift economy, it is a matter of give and take	138
4.7 The gift economy is a family affair	140
4.8 The ultimate kick known as “flow” is the clincher	142
4.9 Why open source is a rush	146
4.10 Conclusion	147
<b>5 An open source culture is better</b>	<b>151</b>
5.1 Open source: a culture in which everyone seems to be doing something	153
5.2 “Merit” is crucial and users are involved	157
5.3 Towards an open source culture in the software development sector	158
5.4 From software engineering to agile/open source software development	161
5.5 Software engineering, components and the human dimension	164
5.6 Open source software development in companies	166
5.7 Differences among traditional, open and agile software development	168
5.8 Conclusion	170
<b>6 How open is the future?</b>	<b>173</b>
6.1 Public versus private	173
6.2 Neither mine nor yours, but public domain	175
6.3 Permission culture: “open” under attack	181
6.4 Private plus public in the software sector	185
6.5 Conclusion	187
<b>7 Thirteen lessons for open innovation</b>	<b>189</b>
Index	195



# Foreword

## Open Up to the Crowd!

I was working for Red Hat in 2001. We were on a corporate golf day with important clients and doing our fair share of drinking the sponsor's beer. It was on the sixteenth hole that I made the very presumptuous statement that the "premium" beer we were drinking was overpriced, badly marketed and tasted even worse, and that I could do better with my eyes closed for next to nothing. On that very hole at Albert Park Golf Course in Melbourne Australia the course of my life changed forever.

After spouting to all who would listen that evening, I awoke the next morning with a heavy head and even heavier challenge: How do I get some customers (who don't exist), to demand my beer (which I haven't invented yet) from my company (which they've never heard of) without spending anything on advertising or traditional marketing, without any experience in the industry, without a brewery or any distribution lines, and yet be profitable?

Of course the solution was all around me. People. All I had to do was set up shop in my customers' minds! Let them have ultimate control – let it be their beer company, not mine! They make the decisions – I just enact their instructions!

I had seen the Linux Operating System start that very way, with the kernel being thrown out to millions of programmers around the world to improve, stabilize, and ultimately commercialize through guys like Red Hat. A tiny drop of water makes no dent on the landscape, but millions of raindrops together can raze mountains. The task for me was collecting the "raindrops" and getting them all flowing in the same direction.

The other challenge was that no one had done this type of crowdsourcing outside of the software industry, and certainly not with any tangible goods. Crowdsourcing as a term hadn't even been invented.

So over the next two years I used a fictional beer I called Blowfly as my "petri-dish" to experiment with concepts and ideas and study the results. Starting with a database of one hundred and thirty friends and relatives, I just threw it at them! Of course eighty five percent of that blew up in my face, but the experiments that worked became the basis of our business today. It was a marketing experiment that went horribly right!

From this came my "7 Pint Beer Plan": many of the points/pints listed to some degree are part of all successful crowdsourcing and open source

business models, and many more are contained within the marvellous book you are reading now.

**Pint 1. Do It Wrong! Rewrite Rules of Engagement**

Understand the rules of engagement in your industry then break them all! If it feels wrong/uncomfortable, you're on the right track! We built our business backwards. We sold a "virtual beer," got potential distribution, and only then worked our production to meet demand. We lowered costs of acquisition and met customers' needs that were not being met by the incumbents.

**Pint 2. Polarize: Be Controversial**

If no one hates you then no one loves you. Rejection is not your biggest threat to success – indifference is. Stop trying to get every customer. Focus on the ones that love you and are emotional about your product or company.

**Pint 3. Lift the Kilt and Let Them Have a Fiddle**

Get Hijacked! Let your customers behind the scenes and get them involved in the development of the product and strategic direction of the company, then ACT on their wishes. Give them both emotional and financial ownership if you can. We gave all our members who voted a right to a share in the company were we ever to go public in the future.

**Pint 4. Embrace Ambivalence**

Look at your business upside down and back-to-front all the time. Businesses are told not to be ambiguous but consistent, clear cut and with fixed positions. This is increasingly untenable in today's profoundly paradoxical world, so embrace ambivalence and you will foster a larger, more robust and stable business through your customer base.

**Pint 5. Embrace "Incongruent Consistency"**

Pop up where you're least expected. You will stand out more, attract more eyes, and if the messages and intent of the business are right, you will attract the people you're looking for.

No beer company has ever advertised in a premium fashion magazine, but we did. We've done so many incongruent stunts and activities that people now expect the unexpected of us.

**Pint 6. Give Recognition**

Forget technology, Viral, Buzz Programs, Affiliates, SEM *etc.* They're all secondary and all change at light speed. What doesn't change and hasn't for 5,000 years is human psychology. Know what makes them tick and build your business towards it. Build curiosity. Build a community of like-minded folk and recognize them publicly when appropriate. Look around

at the people clamoring to be famous. We all love recognition as a race, so use it in your crowd.

#### **Pint 7. Suck Less**

This is not confined to open and crowdsourced businesses *per se*, but it is a rule that is essential for success. Business and self-help books are generally about being better at something. Problem is, we don't know how to be. We have bad habits; we focus on the negative (what we don't have or what we are not). So why not use that as a tool? At Brewtopia, we don't try to be better, just less worse than our competitor, less worse than we were yesterday. If we are five percent less worse in three areas of our business, the effect is compounded by more than the fifteen percent in those areas parlayed.

We never had a book to tell us how to be "Open for Business," like the one you have before you, when we started Brewtopia. But I was there at the bleeding edge, developing robust tenets and principles. So from first-hand experience I have discovered that, when applied correctly and with diligence, the practices and lessons described in the next seven chapters will certainly help you in setting up successful open source and crowd-sourced businesses.

*Liam Mulhall*

CEO

Brewtopia Limited



# 1 The relationship between open source and open innovation

This book explains why and how “open source” and “open innovation” are converging. In the context of software development, we know of open source as an inspired manner of collaboration among motivated individuals working in “communities.” As a concept we know specifically from product development, open innovation is an extension of traditional R&D. The new term for the convergence of both is “crowdsourcing”: an appropriate marriage of the terms “(out)sourcing” and “crowd surfing.” What this means is that “The productive potential of plugged-in enthusiasts is attracting the attention of old-line businesses. It doesn’t matter where the laborers are, as long as they are connected to the network.” This is the way Jeff Howe puts it in his groundbreaking article “The Rise of Crowdsourcing,” which appeared in the June 2006 edition of the monthly magazine *Wired*. The central concept of crowdsourcing will be discussed in further detail in Section 1.4.

Traditionally, R&D is the source of innovation. Companies, particularly large ones, have always painstakingly conducted R&D on their own. They sought patents on inventions in order to introduce their own unique products to the market. If necessary, and often only after a great deal of time, licenses were issued to cash in on discoveries. If market standards were involved, as in the case of the DVD, collaborations followed, and ultimately consolidation into competing consortia. Nowadays the situation remains more or less the same, although it is even more involved. Besides the company’s own R&D as a “source” of “innovation,” we are running into “open source” and “open innovation” ever more frequently. In addition, “open source” and “open innovation” are merging and consolidating. In both cases, this “openness” is based on an unusual form of collaboration that contrasts with the traditional way of doing things.

Open source (open source code) derives from software development, where in-house (proprietary) development had been the norm until recently. Open innovation originated in product development (along with the associated services) and involves a stronger collaboration between companies, knowledge institutes, and innovative users. For both open source and open innovation, the PC, internet, and the appropriate collaborative and simulation software are essential. The supporting software comprises a mix of readily usable multimedia instant-messaging, email, groupware, workflow CAD/CAM, *etc.*

The steadily increasing integration of software into human activity and, consequently, into business innovation explains the growing entanglement of principles and best practices from software development with those of open innovation for product and associated service development.

### **Converging openness**

Every player involved in the purely proprietary software development of ten years ago now embraces open source as a useful supplement to traditional software development practices. And every traditional R&D department is currently participating in open innovation. In the midst of all this activity, many flourishing communities and companies are now adopting a mixture of practices from open source and open innovation. Sometimes this integration occurs in connection with marketing (there is nothing wrong with that, as marketing is an essential ingredient for success), as exemplified by open source shoes. Other times, we find clearly altruistic initiatives, such as in the field of open source pharmaceutical research. Community spirit is an important common denominator in both open source and open innovation. The views concerning commercial utility are quite divergent, however. Obtaining market share is sometimes the driving force behind a “free” policy. But on the more conservative R&D side of things, clever handling of patents still plays a large role. Now and then, this means that software companies release patents and programming. They do this in order to curry favor with customers and with the open source communities in question, and often also to be rid of high ongoing development costs and to acquire better software versions more quickly. In product-manufacturing sectors, Proctor & Gamble is just one of many companies that have, for a little while, been practicing a much less protective patent policy than they had in the past, and reaping a great deal of profit as a result. This convergence of “open source” and “open innovation” promises to be an important feature of our future.

### **Inspiration from open source**

The aim and the core of this book is to draw inspiration from developments in open source software and apply it profitably to business innovation. Anyone unfamiliar with the production of open source software will surely dismiss this as a wild fantasy. After all, innovation is typically fraught with the danger of over-enthusiastic waves of freedom and euphoria. It would seem to be the last thing to which the ostensibly uncontrolled model of open source software production ought to be applied.

At first sight, viewed from the perspective of the traditional organization, the communal character and the non-commercial approach of many open source projects may look suspiciously unruly. But reality proves otherwise. IBM has estimated that its software production could become 30 percent more efficient by cultivating an open source culture. This culture can only thrive when software is constructed in modules. The relationship between the forest and the trees must be crystal clear, or else developing communities may lose their direction and personnel may be unable to ensure the continuity of their work.

In fact, open source means ensuring that the talents of programmers, testers, *etc.* are properly utilized and that the team functions at its highest level. In trendsetting open source projects, such as Linux, Apache and JBoss, there are leaders who make decisions; there is clear communication; goals are made explicit, and there are often well-described processes and procedures. Messing around is not really allowed. Every improvement is evaluated, and if there is disagreement on some point, it is clarified in unmistakable terms. What counts is the result. As a consequence, open source does not mean an uncontrolled and unruly process; it is synonymous with precise knowledge of what you are doing, and with teamwork in which people know their role and understand their place.

Competition among the members of an open source community to record the best result strongly affects the quality of the software produced. The 23 percent growth of Linux on the server market in 2005 (compared to 4.4 percent market growth) would not have been possible without solid and transparent production management.<sup>1</sup> Nearly three quarters of all web servers and therefore nearly all internet traffic operates on open source software. Due to the success of the open-production model in the software industry, it has become worthwhile to investigate the possibility of using such a model for other business innovations.

## 1.1 Open source has many faces

This book will deal with all the special characteristics of an open source process. The most basic element is emphasized: the fact that expert users guide the innovation and production of software, and therefore, there is no clear distinction between producer and consumer. Enhancements by expert users can immediately be included in new product versions, a possibility that is obviously inherent in the intangible nature of software.

Elements of open source are also found in other segments of organizations and of entire sectors. None of these are as far advanced as in the software industry, where open source has completely altered the market. Still, a beginning has been made; many promising examples exist, and expectations are that this trend will continue.

This book will show how open source has changed the software sector, what the principles behind it are, and how they are being applied elsewhere. The intention is to learn from these examples and to re-examine the conventional process of innovation, as the continued spreading of open source culture outside of the software industry has immediate consequences for the manner in which products and services are created. A new type of innovation is emerging, inspired by what has already happened in the open source software movement.

## **The power of open source**

This book could have started by mentioning that NASA dared to send its Pathfinder to Mars with a strong dose of open source software in critical onboard systems. Problems with the software would have meant an irreversible failure for the entire project.<sup>2</sup> Wouldn't NASA have taken only the minimum of risk with such a billion-dollar project? Evidently, open source software is capable of working at the highest level and, hence, does not need to be defended.

"Open" is not opposed to "closed." Open source has now been accepted throughout the software industry and embraced by all "closed" participants, including Microsoft. Still more importantly, the success of open source in the IT sector has been simply a question of overcoming initial opposition. Combining the advantages of "open" and "closed" has made the industry stronger. Open source software has proven itself. Open source culture, the lifestyle surrounding the manufacture of open source software, is emerging as an intriguing option that is attracting attention outside the software industry. An open source manner of doing things, in any of its various manifestations, is becoming more the rule than the exception. Consequently, it comes as no surprise that this open manner of producing software was a source of inspiration for a more general openness: open innovation.

This book is not primarily concerned with what can now be done with open source software. It is not focused on any end product, be it an operating system such as Linux, ERP software such as Compiere or a database such as MySQL. We are interested in the principles underlying open source: how it came into being, what motivates users to take part and how such classic market players as Microsoft and IBM are having to deal not just with open source communities but with an open source culture inside their own organizations. The core of this book is encompassed by the words "business" and "innovation." Our theory is that the extraordinary impact of open source on the software industry will also spread to other economic sectors.



## Innovation à la Henry Ford

Open source software has actually become rather commonplace, if we consider the quality of the products, the services surrounding them and the growing market share. The special significance of open source stems from the enormous change that it has produced and still produces in the software industry. Rapidly growing new companies have surfaced, along with new partnerships between traditional and purely open source players, as well as open source initiatives undertaken by the traditional players themselves.

In software development, market innovation goes hand in hand with open source. In California, Professor Steven Weber, whose name will appear several times in this book, describes open source as innovation à la Henry Ford. Open source also has something to do with the manner in which things are produced. The product itself (be it a car or, as in this case, software) is perhaps not so innovative; it is the way of making it that represents the true innovation. In his book *The Success of Open Source*, Weber puts it this way:

"If I was writing this book in 1925 and the title was *The Secret of Ford*, the argument would be about the factory assembly line and the organization of production around it, not about the car per se. Once we are talking about a production process, a way of making things, then the story is potentially of much greater importance than simply to a few specialists. Processes spread more broadly than do artefacts. Toyota pioneered lean production in a factory that made cars. 20 years later this way of making things had spread broadly throughout the industrial economy. Similarly, open source has proved itself as a way of making software. The question becomes, what are the conditions or boundaries for extending open source to new kinds of production, of knowledge and perhaps of physical (industrial) goods as well?"

Steven Weber, 2004<sup>3</sup>

There springs to mind one obvious similarity between the new production method pioneered by Henry Ford and the way in which open source software is created: the industries affected by these changes were and are radically altered. The assembly line and mass production sparked great change in the auto industry. Cars became so cheap that "the common man" could afford one. The software industry has also been radically transformed by a method of production. This production process and its characteristic open source culture are forcing software producers to reconsider their strategy and manner of working.

## 1.2 Seven characteristics of open source

Before going into the lessons that can be learned about innovation from open source, let's first consider what open source is, exactly. It has seven distinct characteristics, starting first with the fact that it is the users who take the lead in innovation. Three individuals deserve special consideration in this regard. The first is Richard Stallman, the person who actually started it all. Frustrated with the fact that software users were prohibited from accessing the software code because companies wanted to make as much money as possible from software, he founded the Free Software Foundation ("free" in the sense of "liberated").

The movement developed a new type of copyright to protect users who wanted to participate in software innovation. Later, Brian Behlendorf, the second person on our list, worked with a number of others to develop Apache webserver software, one of the most successful open source software products to date. Behlendorf then used his knowledge and experience to launch a collaborative platform called CollabNet: hundreds of thousands of people now work on open source projects on CollabNet. Number three on the list is Eric Raymond. He smoothed out the rough edges in the movement started by Richard Stallman, producing refinements that enormously stimulated the further commercialization of open source.

### **Nota bene**

*The seven characteristics mentioned below are closely related to the thirteen lessons of open innovation that collectively constitute the concluding chapter of this book. The reader could simply adopt all twenty without going any further, but that would miss the case histories and the analysis contained in the five intervening chapters. Instead, the impatient reader could obtain an overview by reading Section 1.4, which briefly sketches an outline and describes the structure of this book.*

### **Characteristic 1: Special licenses ensure that everyone can access the source code; they also protect the community of developers**

A persistent malfunction in the paper feed of a laser printer in the Artificial Intelligence Laboratory at MIT played an important role in the genesis of open source software. A programmer named Richard Stallman wanted to resolve the problem and requested the source code from Xerox, the supplier. At the beginning of the eighties, it was not unusual in the academic world for users to correct apparent flaws in the freely accessible software source codes. However, this time Xerox refused to provide the source code. Stallman, however, who felt that users had the fundamental right to learn from software and to create new things with it, opposed the Xerox stance. He was concerned that the source code of software to which a community

of users had made important contributions was no longer communally available. In 1985, Richard Stallman set up the Free Software Foundation (FSF). Its purpose was – and still is – to protect the rights of software developers who wanted to work collectively to make software and neutralize the risk of copyright claims by third parties. Therefore the FSF developed a new type of contract called the General Public License (GPL). This became commonly known as “copyleft” in contrast to the standard “copyright.” In effect the GPL protects the rights of software makers while allowing users to freely build upon the products. Free software began to flourish – software delivered along with a source code that everyone could learn and use to create new versions, or even new software. It was a breakthrough when Linus Torvalds obtained a GPL for the Linux operating system in order to reassure “his” Linux community that whatever they did would remain freely accessible to everyone. Various types of GPL are now the official bonding agreements for participation in open source communities.

### **Characteristic 2: Users take the lead in the development of new software**

At the beginning of the nineties Brian Behlendorf, a young student at Berkeley, was tinkering with the web servers in his business school’s computer lab. His skills were recognized by some of the people at the magazine *Wired*, who subsequently invited him to set up the very first online magazine: *Hot Wired*.

For this website, Behlendorf used an open source program from the National Center of Supercomputing Applications (NCSA), but it did not serve all the needs of the *Wired* website. Accordingly, he made some adjustments, and others did the same for their specific purposes. These patches (hence the name “Apa[t]che”) were sent back to the NCSA, which wasn’t able to implement and support all the changes. Therefore, Behlendorf decided to take control himself. Working with a small group, the original NCSA code with all the patches was compiled and rewritten as the first edition of the so-called “Apache” webserver. Behlendorf later revealed that the co-ordination of this software project was primarily based on progressive insight and volunteer contributions of everyone who had the inclination to take part. The users of the webserver software continued to develop new releases. Later, established software suppliers came out with their own webserver programs, but Apache has remained the industry leader.

### **Characteristic 3: The internet and virtual workstations are the driving force behind open source**

The dissatisfaction of Behlendorf and others ultimately led to the software of the Apache webserver. It currently dominates the market with

about a 70 percent market share. Not only is software development a community enterprise, the community also provides user support. As Behlendorf and the other developers were unable to provide ongoing technical support, the users began to organize themselves. As it turns out, this “free user-to-user assistance” functions remarkably well. Research by Karim Lakhani and Eric von Hippel<sup>4</sup> shows that a commercial contract is not at all required to obtain good help for questions and problems. The internet and virtual workstations enable users to assist each other effectively.

Later, Behlendorf established the virtual workstation CollabNet. It is a website where people who want to work on open source projects can meet and collaborate. The number of registered CollabNet users has grown to more than 700,000. Another virtual platform for software development called SourceForge numbers over one million users.

#### **Characteristic 4: Self-determination and intrinsic motivation are the dominant open source principles**

Web environments like CollabNet only represent the technical side of the business. What motivates people to make something like Apache is an entirely different story. As already noted, voluntary and unpaid contributions by individuals have produced high-grade software products such as the Apache webserver, the Linux operating system, the MySQL database and the JBoss application server. New software is “created” without explicit contracts or payment. Self-direction by people who want to participate without direction from a boss, as well as strong intrinsic motivation without the need for monetary reward, makes up the “soft” power behind the open source software movement. This self-management ensures that the expertise of participants in open source projects is employed in the best possible manner. Everyone can select a personal challenge, which naturally leads individuals to perform work in which they excel and to win recognition for it. The intrinsic motivation is primarily nourished by the work itself. The extent to which software development is found to be inspiring compensates for the lack of income. Furthermore, a reputation built up in an open source community often increases the chances of obtaining lucrative employment.

#### **Characteristic 5: Open source aims at commercial success**

Not everyone is happy with the moralistic attitude of the Free Software Foundation. A number of parties are eager to bring open source products to the market, but the GPL license sometimes has an obstructive effect. This was why Eric Raymond set up the Open Source Initiative (OSI). The

term “open source software” was introduced in 1998. Gradually, “open source” replaced the term “free software,” which had been widely used up to that time. The accent has now shifted from “free” to “open” because free also means “for no money,” a non-commercial implication that the profit-driven open source movement strives to correct.

Whereas the word “must” occurs frequently in the definition of GPL licenses, we find the word “may” more often in the definitions of OSI licenses. Software that satisfies the conditions of the OSI “may” be called “open source software” and is given a place on the OSI website. In any case, the following three points are applicable, according to the rules of open source software:

1. The source code must be provided or made available for a price no higher than the distribution costs.
2. Everyone is permitted to redistribute the software without paying any royalties or licensing fees to the authors.
3. Everyone is allowed to modify the software and to redistribute the modified software, to which the same conditions apply as are applicable to the original software.

### **Characteristic 6: Management by transparent and meritocratic leadership**

Although “free” and “open” might suggest otherwise, the production management of open source is carefully orchestrated. In open source projects, there is a strong focus on rigorous procedures and a clear division of roles. Sometimes decisions are made by holding a vote, as is the case in the Apache community. There are completely unambiguous guidelines for resolving conflicts. To sit on a project management committee, you must first be invited. These people are granted voting rights and jointly determine what is the best solution. It is even possible to speak of a “meritocracy”: authority is allocated on the basis of merit. Anyone participating in an open source project is awarded greater standing and has greater say. Just as in ordinary organizations, the open source community has its heroes and leaders. For example, Linus Torvalds has this status insofar as the Linux kernel is concerned. He heads that particular open source software project. Under him are his “lieutenants.” It is absolutely clear how decisions concerning the Linux kernel are made. Descriptions of the processes and procedures of open source projects can be found on relevant internet sites. Another important characteristic of the clever management of open source projects is openness to feedback and mutual criticism. Even the meritocratic leaders are sometimes exposed to fierce criticism. The open production method is characterized by this attitude, which is the key to the method’s success. Everything revolves around the code

and the quality of the work. Political games are slightly less easy to play in the open source world.

### **Characteristic 7: The success of open source has generated a hybrid model**

Since 1998, the entire free/open source landscape has been transformed into a mixture of both closed and open source commerce. Large commercial software companies have become progressively more amenable to open source. Salaried employees at Sun Microsystems, IBM, Microsoft, Oracle, SAP and HP have been instructed by their employers to collaborate in open source communities. The free and open source licenses are increasingly used for what was once closed software produced “in house” and which has now become available as open source. For example, Sun has made its Solaris operating system available under an open source license, so the original thousand developers now collaborate with ten thousand programmers and testers in the open source community. IBM has now fully embraced the open source production model, with self-determination and intrinsic motivation as a guiding principle. This open source culture is seen as a way to improve a company’s productivity by at least a third. Under the labels of “common source,” “shared source,” and “community source,” the open source concept is spreading throughout the closed software factories of the commercial software giants. At the same time, an increasing number of open source companies are working with two licenses: an open source license and a regular commercial license. Little seems to remain of Richard Stallman’s idealistic free-software philosophy, but we should not forget that many open source products – Linux, for example – are still available through a GPL license.

## **1.3 Open source inspiration for innovation**

The characteristics identified above – user involvement in innovations, a new type of license, unpaid work, a community of strongly motivated participants, transparent and meritocratic management, closed and open models that supplement each other – are all aspects of open source software. However, these features were not generated by and are not unique to the open source software movement. For instance, good companies often have exceptionally motivated people among their employees. Unpaid work has been around for ages, in the form of various volunteer activities. Not even the integration of open and closed models is a new phenomenon, as there is hardly any company that undertakes any form of modernization purely on its own. And creative licenses are not only associated with open source software.

The most important innovation does not involve the factors themselves but rather how they are combined in open source software. This success, the success of open source software, should appeal to every business manager. The natural expectation is that this success, the combination of factors that makes open source such a compelling phenomenon, is also applicable, in one way or another, to other fields: inside organizations striving for innovation, but also outside of them in economic sectors other than the software industry.

## Open innovation

With open source in the subtitle of this book and the link being made to innovation in general and not only in the software domain, it is tempting to use the term “open innovation.” Essentially, what we call “innovation” here may just as well be called “open innovation.” Therefore, there is “open source” and “open innovation”; note, however, that the two don’t always mean the same thing. Henry Chesbrough, affiliated with the University of California and Director of the Center for Open Innovation, published the standard book on open innovation. The expression “open source” does not occur in his book and nothing is said about the opening of innovation by involving users in the innovation process. However, there are certainly important parallels, because the notion underlying open innovation as viewed by Chesbrough not only emphasizes the need for an outward-looking stance but also urges companies to abandon the idea that they must keep all knowledge to themselves in order to distinguish themselves in the marketplace. This is connected to the management of a company’s intellectual property, which demands better and more clever strategies, if only on account of the cost required to acquire and maintain patents all around the world. Nevertheless, it is the openness generated by open source software development that we are concerned with, and which provides us with a starting point to refine the definition of the concept “open innovation.” It involves managing intellectual property and attracting (expert) users, and just as in open source software production, it is a manner of adopting an outward-looking stance. In 2006 Chesbrough published *Open Innovation: Researching a New Paradigm*<sup>5</sup> in conjunction with Wim Vanhaverbeke and Joel West. Patterns of open innovation in open source software are also discussed in that book.

Along the same line, the influential Digital Connections Council (DCC) of the Committee for Economic Development ([www.ced.org](http://www.ced.org)), a business and university-led public policy group, issued *Open Standards, Open Source, and Open Innovation: Harnessing the Benefits of Openness* in April 2006.<sup>6</sup> This report examines open standards, open source software, and open innovation. The report concludes that “openness” should be promoted as



a matter of public policy in order to foster innovation and economic growth in the U.S. and world economies. The DCC is chaired by Paul Horn, senior VP Research at IBM.

In contrast to closed innovation, and keeping the seven characteristics of open source in mind, classic open innovation gives us insight into how both forms of openness might reinforce each other. A table taken from a presentation from Philips, extracted from the 2004 consultancy report entitled *Vitalizing the Knowledge Economy (Vitalisering van de kennis-economie)* makes this point clear. The view is derived from the Dutch Innovation Platform by Herman Wijffels and Thomas Grosfeld.<sup>7</sup> In the vitalization report, open innovation is included along with three other factors: “internationalization,” “knowledge economy as an economy of learning,” and “multidisciplinary ways of approaching research and problem resolution.”



**Closed innovation**

The brightest people in our field work for us.

To profit from R&D, we have to invent, develop and produce for ourself.

If we invent something ourself, we will be the first to bring it to the market.

**Open innovation**

Not all the clever people work for us: we must find a way to tap into these other human resources.

R&D by others can provide significant added value. Our own R&D is necessary in order for us to apply a part of this added value.

Research does not have to be ours in order for us to profit from it.



Closed innovation	Open innovation
The first to announce an innovation is the winner	A good business model is more important than being the first to market.
Whoever has the most and the best ideas, wins.	If we make the best use of internal and external ideas, we will come out on top.
We have to protect our intellectual property so that other companies cannot make any use of our ideas.	We must profit from the use of our intellectual property and we must use the intellectual property of others that suits our business model.

Here, open innovation is clearly presented in a better light than closed innovation. The point at the top of the right column in Philips's table ("Not all the clever people work for us; we must find a way to tap into these other human resources") is key to an open source approach. It entails making good use of the internet and inviting users to participate, in order to take advantage of all available expertise and experience, and therefore to profit from intrinsic motivation and self-determination.

Consequently, the central principle of open innovation is and remains making clever use of knowledge outside the company's own doors and possibly from important users of the products as well. If you speak to Philips, you will hear that this company has been busy for decades with patent management and collaboration with partners in order to operate more effectively. So Philips was long ago practicing this form of innovation, but has now also plotted a clearer course insofar as the involvement of end users is concerned, as demonstrated by the following newspaper clipping:

#### Philips experiments with consumer-led innovation

Producers who are open to suggestions from lead users [or innovating consumers, ed.] can improve their products faster, states MIT professor Eric von Hippel. Philips puts the theory into practice by means of the website Leadusers.nl, which has been operating out of Eindhoven since August. The company has already used the website to conduct research into video telephony and is currently undertaking a study on the quality of sleep.

Emerge, 2006<sup>8</sup>

The Dutch Science and Technology Policy Council AWT (*Adviesraad voor het Wetenschaps- en Technologiebeleid*, AWT) is clear about what is meant by open innovation, and endorses our position: it is the acknowledged open innovation practiced for decades by such companies as Philips, together with the involvement of users in innovation.<sup>9</sup>

### **The central principle of open innovation according to the Dutch Science and Technology Policy Council, AWT**

The shift from closed to open, collaborative, innovative processes means assembling various types of knowledge to form chains and networks. This not only involves technical and social-scientific knowledge but also and primarily the experiential knowledge of end users. Therefore the innovation will have greater relevance to services, experiences, and users than to products alone.

In the Netherlands and elsewhere around the globe, open innovation in its various guises is a major issue. In December 2006, the Dutch Ministry of Economic Affairs, in conjunction with the Organization for Economic Co-operation and Development, OECD, held an international conference on the topic, in the context of globalization. The conference was attended by Henry Chesbrough, Wim Vanhaverbeke and Eric von Hippel, all of whom were mentioned earlier; Erkki Ormala, who acts as Nokia's Director of Technology Policy and is a former Secretary of the Science and Technology Policy Council of Finland; Yutaka Yoshimoto, a Chief Representative in the Paris Office of the Japanese Ministry of Economy, Trade and Industry; and last but not least the Dutch Science and Technology Policy Council AWT, represented by Pieter Adriaans. This clearly shows that open innovation is a hot topic on the economic agenda in many different countries.

Open innovation and open source mirror each other well. The Dutch Science and Technology Policy Council, AWT, identifies precisely those enterprises in which open source software is prominent as conforming to what Chesbrough conceives as open innovation. And the possibility of including end-user experience has grown enormously in recent years: first of all due to the internet, which facilitates contact more easily and allows others to participate in innovation; and second, because more and more people are proficient and eager to be digitally active. As a result, it is increasingly easier for users to participate.

The rationale behind open innovation is to have all intelligent people working for one company. This is represented on the left side of Figure 1.1 and is taken directly from Philips's table. Open source software provides the building blocks by which more smart people can become involved in the organization. On the right are a few characteristics of open source that fuel innovation with an impulse to actually involve the necessary expertise. The diagram suggests that attracting this involvement becomes increasingly easier. This is also the reason why it is possible to speak of an escalating trend. Technology makes it possible, and ever more people are willing to take part. This form of open innovation would not have been possible if internet use had not become so prevalent worldwide. The diagram also reveals the tension that has been created involving property rights. Patents and the protection of intellectual property belong to tra-

ditional approaches to innovation, on the left side. Obviously, companies want to earn as much money as possible from their discoveries. But if these discoveries increasingly come from outside the company, friction results. After all, who can claim to be the owner of a good idea: the company or “the community” that produced it? In legal terms, “copyleft” means that no exclusive rights are permitted and everyone has the opportunity to earn money from collectively developed ideas. The fact that money can be made in this way is demonstrated by open source software. Specifically, companies that can add value to the software acquire paying customers. The two forms of property rights are therefore compatible and each can be used to earn revenue. Nevertheless, a tension exists between them, a subject to which we will return in Chapter 6.

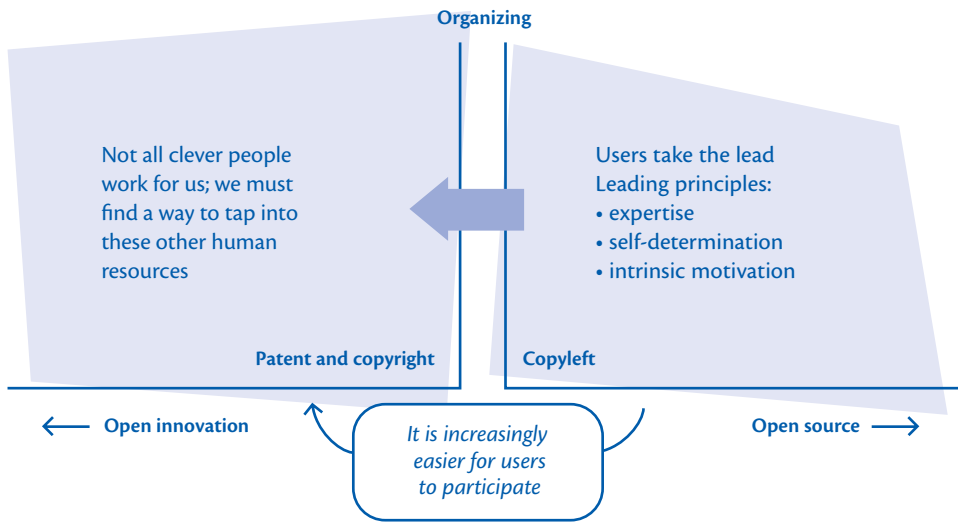


Figure 1.1: Open source revitalizes classical open innovation

### 1.4 Crowdsourcing

In the context of open source, open innovation and user-driven innovation, crowdsourcing is a new general label encompassing the specifics of the other terms. Crowdsourcing is all the rage, especially at the end of 2006 now that *Time Magazine* has chosen YOU as the person of the year.<sup>10</sup> Peer-to-peer communication is the leading principle behind this movement, and the modern internet the motor powering it.

The term “crowdsourcing” suggests, above all, that organizations ought to build bridges to communities and actively collaborate with them. Crowdsourcing is shared activity! Additionally and perhaps more importantly, crowdsourcing frees us from the stigma attached to free and open-source-inspired innovation: the notion that such innovation only involves

open source software. Of course, this is not the case. Crowdsourcing can be performed by any organization in any sector, with no need for open source software. Only the production principles are the same.



### Open for Business

Currently, the merging of innovation with open source software is certainly very relevant. For instance, the title of this book, *Open for Business: Open Source Inspired Innovation* was born out of an extensive open source project called “Open for Business.” The Apache Open for Business Project<sup>11</sup> is a software suite complete with all the trimmings, including ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), e-business/e-commerce and SCM (Supply Chain Management). Innovation in an open source environment is therefore becoming increasingly easier.



All well and good, but a greater effort certainly needs to be made in order to reach a broader audience. Beyond the circles directly and professionally involved in open source and community source, or in open innovation<sup>12</sup> and user-driven innovation,<sup>13</sup> this type of innovative activity has been fraught with difficulty.

## Crowdsourcing

This all changed in June 2006. That is when the term “crowdsourcing” suddenly surfaced in *Wired Magazine*.<sup>14</sup> At that time, we were researching the original source of crowdsourcing, open source software production. Just like Jeff Howe, we had noted that open source production principles were spreading to other economic sectors: early instances of crowdsourcing.

In the meantime, crowdsourcing was becoming the title under which all types of open-source-related innovation were being categorized. “Crowdsourcing” brings a whole array of openness, ranging from concept development to production, under a single overarching concept that can be immediately and intuitively understood.



Jeff Howe, the editor of *Wired*, was the originator of the term “crowdsourcing”<sup>14</sup>

“Crowds” are internet crowds, and they can collaborate intensively without crowd members actually being in close proximity or even online at the same time. In fact, being online simultaneously is often convenient but, in most cases, not absolutely necessary.

Of course, not every internet crowd is made up of thousands or even dozens of people; just think of the expression “Three is a crowd.” In reality, crowds are just indefinite numbers of people located somewhere in the world. And needless to say, many cases or process phases are subject to the principle that the greater the obsession, the greater the productivity – especially where the testing of software components is concerned.

“Indeed, there’s nothing wrong with a buzzword if it actually signifies a meaningful trend or development.”

Jeff Howe, *Wired Magazine*

## Innovation happens elsewhere

Ultimately, anyone wanting to tap into the public's creativity and having given appropriate consideration to setting the process in motion can just allow crowds to work on their own and see what comes of it. The Cambrian House software company operates in such a manner. You, the crowd member, do the thinking, programming and testing. And you, the crowd member, have a large share in the profits as well. Selection and sales is what the company is especially good at<sup>15</sup>. The resulting collaborative product is what they call "crowdsourced software." The software industry has been on to this for twenty years, only now the crowd is sharing directly in the profits.



However, it is also possible to choose to have greater control and secrecy. There are all kinds of variants, although the dominant mantra remains, "Innovation Happens Elsewhere."<sup>16</sup> Being open to this outside innovation essentially requires you to look beyond your immediate vicinity, to use the internet to facilitate collaboration among the right fanatics and to allow them "to do their own thing."



*Innovation Happens Elsewhere: Open Source as Business Strategy* is an important book by Ron Goldman and Richard Gabriel published in 2005<sup>16</sup>

Typically, such practices range from blogging<sup>17</sup> and YouTubing<sup>18</sup> to paying a few cents to use Amazon's Mechanical Turk<sup>19</sup> to perform a few tasks; from drawing craters to help NASA determine the age of areas on Mars<sup>20</sup>

to guided efforts such as the Global Innovation Jam<sup>21</sup> at IBM; from developing the Wikipedia online encyclopedia<sup>22</sup> to assisting in the development of medicine as part of the Tropical Disease Initiative<sup>23</sup>; and from the Apache Open for Business Project<sup>11</sup> to user-driven innovation<sup>11</sup> at Philips, 3M and other organizations.

It is remarkable how many people are active on the internet these days, in very different crowds. They are working together for a good cause, for science, for a company, purely for money, or in any number of pursuits, including developing faster and better innovation for companies and institutions.

## Peer-to-peer e-mancipation

Peer-to-peer (abbreviated p2p) is a buzzword that is often heard in this context<sup>24</sup>. Literally, p2p is the direct e-communication of one individual “peer” (like or like-minded person) with another. Such person-to-person contact within the internet population must not, however, be confused with a complete p2p economy.



Anyone can begin a second life on the internet by becoming active in communities, for example in “Second Life” itself

The internet certainly makes it easier for us to find others with certain common and complementary interests. Market implications are certainly not irrelevant: quite the contrary. The fact that, in many instances, p2p communication bypasses or partly bypasses traditional intermediaries means, first of all, that the market works more efficiently in terms of lower costs and faster delivery times. But it is also more effective since, in



principle, increasingly more individuals can be served on demand. In fact, p2p stands for the increasing internet involvement of the individual and of the many widespread like-minded people who are co-present. In a word, it stands for an “e-mancipation” that more than ever turns the individual customer into the king, especially when such individual kings amass into sovereign crowds.

The significant economic activity associated with this form of crowd rule is perhaps best demonstrated by the well-known example of the Second Life virtual world<sup>25</sup>. To begin with, it could never have been built without the efforts of an enthusiastic crowd. Besides the individuals who acquire all types of virtual commodities, ranging from clothing to real estate, more and more real-world companies are becoming active in Second Life, companies such as ABN-AMRO, IBM, Philips, Nike, Reuters, and Nissan.

## YOU!

With all this in mind, it is not surprising that *Time Magazine*<sup>10</sup> has elevated YOU to a place of honor, just as Business 2.0 had previously done<sup>26</sup>. YOU are an individual who is actively contributing to the emergence of the Information Age. YOU represent a class of people, rapidly increasing in number: all those individuals whom we call “prosumers,” otherwise known as digitally productive consumers.

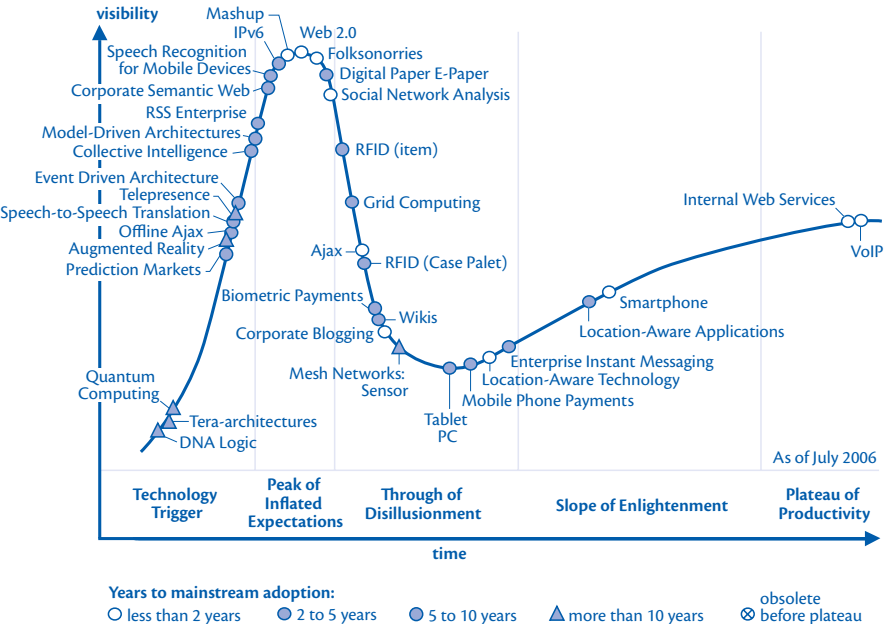
A few of these prosumers, whom *Time Magazine* surely had in mind when it elected YOU as person of the year, are the makers of the Chumby<sup>27</sup>. Take a look at the website. It’s a compact generic electronics device that can act like a clock radio and also exchange photos and messages with your friends. But first and foremost, YOU can add functionality and, of course, change its looks according to your whim. The most important point is that the Chumby does not come from the design department of a large electronics manufacturer. It was simply thought up by a couple of hackers and made by hackers for other hackers, who will subsequently tinker with it further, mainly to enrich their Net life:

“We invented the Chumby because we, like most people we know and especially like most younger people we know, have at least two often-incompatible lives. We have our **real life**: being with our family, friends and co-workers; doing activities; running errands; making dinner. We also, and increasingly, have our **Net life**: answering email; posting to our blogs; sharing digital photos or the latest joke; and just letting Web whimsy wash over us.”

Digitally active individuals and crowds are reforming the world. *Time Magazine* waxes lyrical about this transformation, but the reality is slow



and gradual, complete with ups and downs, as well as a hype-cycle rhythm that we know so well from Gartner<sup>28</sup>.



This hype cycle was devised by Gartner in August 2006 and charts the new technologies of the modern internet

Crowdsourcing is not literally mentioned on the above hype cycle, although “collective intelligence” comes awfully close and occurs near the cycle’s peak. Moreover, crowdsourcing is strongly related to Web 2.0, which is precisely at its summit. The productivity plateau has not yet been reached – or has it? For where would you place Linux and Apache? The software sector is far ahead. A world without crowdsourced software has become unthinkable, and crowdsourced anything is gaining enormous momentum.

### Crowdsourcing and open innovation

We decided to deploy the term “open innovation.” The term is a valuable concept and necessary addition because it allows crowdsourcing to enter the field where the game is being played. Innovation applies to the entire process, from conceptualization to development and use – from “invention” to “cashing-in” so to speak. Open innovation means that companies have to open their processes to third parties. The many examples found in Chapter 2 show how the crowd can provide a notable contribution at

every stage of these processes. Taken together, these examples reveal the manner in which production, R&D and marketing companies are being changed by this open model.

## 1.5 The outline and structure of this book

Following this introductory chapter, we take a closer look at open innovation outside the software sector. Subsequently, we review the successes of the open source movement within the software sector. This market has taken the lead in embracing the open model, with a degree of acceptance that is described in Chapter 3, which is where we highlight the lessons on innovation to be drawn from open source software production. In two more specific chapters we come to the core of open source software, the essential elements that can serve as inspiration for business innovation. In this way, the characteristics of open source are passed under the microscope of this book. In Chapter 6 and in preparation for the last thirteen lessons that, as mentioned previously, are closely related to the seven characteristics of open source in Section 1.2, we pose the important question, How open is the future? Each of these chapters is briefly discussed below. The concluding chapter, Chapter 7, contains the thirteen lessons that are related to the seven characteristics of open source in Section 1.2.

### Chapter 2: Open innovation – the new trend

In the introduction we breezed over the concept of innovation perhaps a little too simply, but we will certainly make up for this in the second chapter with reference to theories by three experts on innovation in order to discuss the issues in greater detail.

The first, Joseph Schumpeter, is the originator of the structured study of innovation and the man who defined innovation as something that must primarily make cash registers ring. The evidence that this is what has happened with open source software is presented in Chapter 3, where we refer to the operations of existing IT companies to show how open source software earns money.

The second expert is Clayton Christensen, a professor at Harvard Business School and the person who called open source a “disruptive innovation.” In other words, open source is a phenomenon that has turned the software market completely upside down. This only makes the innovative power of open source even more relevant because comparable innovations could have a similarly disruptive effect on other markets.

Finally we bring in Eric von Hippel, head of the Innovation and Entrepreneurship Group of the MIT Sloan School of Management. For a quarter of a century Von Hippel has been researching “user-driven innovation,”

the “consumer-led innovation” of which open source software provides an outstanding example. It is interesting that he runs into user-motivated product changes both inside and outside the software industry. He locates the relation between open source and innovation primarily in the bottom-up initiatives of users.

Chapter 2 provides examples of open innovation and shows how its latest generation participates in this model.

Next, the second chapter specifically describes the examples of open innovation “elsewhere” -outside the IT sector. Examples from the media sector, the aircraft industry, pharmaceuticals and the consumer goods industry sketch an interesting picture. In several areas and in various sectors, efforts are made to exploit the possibilities of open innovation and to expand upon them. The chapter reveals that the new generation of users, which is named “Generation C” (for “content,” “code” and “creation”), show themselves to be stronger “open innovators.”

### **Chapter 3: The software market has taken the lead**

The third chapter outlines the game that is played by the software giants who control the IT industry. There is not a single player who is not involved with open source. Even Microsoft, so frequently portrayed as the big opponent of open source, has become linked with a thriving community of developers outside its own company walls. Microsoft fully understands what open innovation can mean. The same cannot be said of many companies outside the software sector. The IT market has, therefore, taken the lead, and its advance can be measured by what the players are doing in this market. What they are all attempting to do is to make money. Joseph Schumpeter has described the true picture right from the start: innovation is extremely enjoyable and exciting, but the bottom line is using it to generate business. Precisely for this reason, the large market-controlling IT manufacturers have gone in for open source in a big way, while relative newcomers, like the open source software company Red Hat, make a decent living as well<sup>29</sup>. Something is clearly going on here. With a little money, visions were converted into activity, and the entire IT sector is now completely immersed in this enterprise. The exact change-over point cannot be clearly delineated, but IBM’s support for the Linux community played an important role. And the release of five hundred patents to the open source community by IBM at the beginning of 2005 marked a new milestone in the history of open source. John Kelly, the senior vice-president of technology and intellectual property called it “the beginning of a new era in how IBM will manage intellectual property.”

Chapter 3 describes how open source has transformed the IT market and shows how this sector is at the forefront of this new open trend. Ultimately, it provides a jarring wake-up call to other sectors concerning the power of open innovation.

We will reveal the steps that a number of large suppliers have taken to make their companies and products more open. Such software giants as Microsoft, IBM and Sun use open source, first of all, as a strategic weapon in order to further dominate the market. We examine how all that fits together. The IT establishment will be lined up and confronted with the “challengers.” Companies such as MySQL, Zend, Red Hat, SugarCRM and other open source enterprises have joined the competition. Where do these companies get the money to enter the market and earn their keep? We will see how the established order collaborates with these newcomers, even by buying them out if necessary.

#### **Chapter 4: What motivates an open source community?**

Chapter 4 investigates one of the most important issues that open source can teach us: what motivates people to participate in the production of software, in most cases without receiving any payment for their work? And the crucial inspiration for this investigation clearly emerges in the subsequent question: what can we learn from this regarding open innovation in other business sectors?

This chapter carefully examines the new generation of developers who have collectively built software products such as Linux. What keeps them going? Why are they doing it? What motivates them?

Chapter 4 considers the most fascinating questions surrounding open source: Why do people do voluntary work? And what does this mean for open innovation outside of IT?

From an economic viewpoint, open source initially seems to be easy to explain. Assuming the “economic viewpoint” means assuming that people primarily undertake activities in order to generate income from them. Consequently, we provide a glimpse into the inner workings of open source communities. It is a mistake to think that people devote time to open source for idle motives, and this notion is swiftly dispelled. Extensive motivational research has shown that most people are made of flesh and blood, so they have some solidarity with each other but also want to make a living. A review of the most important studies will reveal that a mixture of motivational factors makes open source production possible.

## Chapter 5: Open source culture is better

Motivation alone does not generate products. There must be management and at least a minimum amount of focus in order to bring something into production. We explain how such companies as Apache and Red Hat were able to succeed in this regard and how Linus Torvalds manages his Linux kernel. The interesting issue regarding the governance of open source communities is that they deviate from the well-known managerial models in a number of important respects; nevertheless, it is unquestionable that a number of good products have resulted. The question then arises as to why all software is not made in this manner. Could open source software be applied to the manner in which your company is currently developing software?

Chapter 5 reveals the extent to which open production and its associated culture has already become mainstream. The software industry is leading this trend. And the inspiration for applying open innovation elsewhere lies hidden in the lessons that can be drawn from this production method.

An open source culture has a beneficial effect on the quality of software production. Remarkably, this is recognized by everyone, and equally remarkably, the current manner of software production already displays very many open source traits without identifying itself as such. But deliberate efforts to establish and cultivate an open source culture inside company walls have enjoyed little success to date. Walt Scacchi, a member of the Institute for Software Research at the University of California, analyzed this issue. The community-source initiative at IBM in 2005 was a telling event. IBM expects that the official establishment of an open source culture will enable it to develop its software at least 30 percent faster. Microsoft is following this lead. The problems involved in the delivery of Windows Vista when the project was still called ‘Longhorn’ have caused a new wind to blow through the Microsoft software-development culture. Bill Gates gave the order to design more in the form of “Lego blocks” and to work on incremental improvements, such as is usually done in open source projects. Evidently, open source software development and classical “software engineering” have come together in striving to provide new products faster.

## Chapter 6: How open is the future?

Chapter 6 goes beyond the frontiers of “open.” The possibilities of open production seem unlimited, but we run into boundaries that limit the further development of open innovation. The owners of knowledge, soft-

ware and other creative work, whose ownership is secured in patents and copyrights, want to cash in on their creations as much as possible. However, the open innovation model thrives best when all available knowledge is the subject of as much elaboration as possible. Continuously requesting permission to use certain proprietary intellectual property is a tiresome and nearly impossible task. Should we not in fact rid ourselves of ownership protection? Are we not trapped in a legal system that no longer suits this innovative age in which we find ourselves?

Chapter 6 describes how the struggle for ownership is a tension between private property and the public domain. This struggle is critical for the further development of open innovation.

It is far from certain that the future will be significantly more open. There are reactionary forces and they have everything to do with the legal conflicts surrounding ownership. The inverse of open, the closed patented discovery, is certainly not in retreat. Understandably, companies want to make money from their innovative efforts. But paradoxically, protecting the right to earn a buck from a company's own R&D places limits on economic growth. At least, opinions diverge on this issue. The possible results of such conflict will be examined from three points of view: private property, the public domain and a blend of private and public.

What these facets of the legal system certainly make clear is that the established order, both in the software world and other industries, will fight tooth and nail to defend its business model. But what also becomes apparent is that the permission culture in which we live often restricts productivity.

#### Notes to Chapter 1

1. Shankland, S., "Windows Bumps Unix as Top Server OS" *CNET*, February 21, 2006, [news.com.com/Windows+bumps+Unix+as+top+server+os/2100-1016\\_3-6041804.html?tag=html.alert](http://news.com.com/Windows+bumps+Unix+as+top+server+os/2100-1016_3-6041804.html?tag=html.alert).
2. Norris, J.S., "Mission-Critical Development with Open Source Software: Lessons Learned," *IEEE Software*, January/February 2004, [www.ics.uci.edu/~wscacchi/Papers/New/Hold/jpl-MissionCritical-oss.pdf](http://www.ics.uci.edu/~wscacchi/Papers/New/Hold/jpl-MissionCritical-oss.pdf).
3. Weber, S., *The Success of Open Source*, Harvard University Press, 2004.
4. Lakhani, K. & E. von Hippel, "How Open Source Software Works: 'Free' User-to-User Assistance," *Research Policy* 32, 923-943, 2003, [opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf](http://opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf).
5. Chesbrough, H, W. Vanhaverbeke & J.West, *Open Innovation: Researching a New Paradigm*, 2006
6. Digital Connections Council of the Committee for Economic Development, *Open Standards, Open Source, and Open Innovation: Harnessing the Benefits of Openness*, April 2006, [egovstandards.gov.in/news/u-s-advised-](http://egovstandards.gov.in/news/u-s-advised-)

- to-promote-open-standards-source-innovation; [www.ced.org/projects/ecom.shtml](http://www.ced.org/projects/ecom.shtml).
7. Wijffels H. & T. Grosfeld, *Vitalisering van de kenniseconomie: het beter ontwikkelen en benutten van de mogelijkheden van mensen als de sleutel voor een dynamische kenniseconomie*, Innovatieplatform, November 2004, [www.rug.nl/tlg/documenten/VitaliseringVanDeKennisEconomie.pdf](http://www.rug.nl/tlg/documenten/VitaliseringVanDeKennisEconomie.pdf).
  8. [www.emerce.nl/nieuws.jsp?id=1102473](http://www.emerce.nl/nieuws.jsp?id=1102473).
  9. AWT werkprogramma 2005: groslijst van mogelijke onderwerpen, [www.awt.nl/uploads/files/groslijst2005.pdf](http://www.awt.nl/uploads/files/groslijst2005.pdf).
  10. [www.time.com/time/magazine/article/0,9171,1569514,00.html](http://www.time.com/time/magazine/article/0,9171,1569514,00.html).
  11. [www.ofbiz.org](http://www.ofbiz.org).
  12. [en.wikipedia.org/wiki/Open\\_Innovation](http://en.wikipedia.org/wiki/Open_Innovation); [www.openinnovatie.nl](http://www.openinnovatie.nl).
  13. [leaduser.com](http://leaduser.com).
  14. [www.wired.com/wired/archive/14.06/crowds.html](http://www.wired.com/wired/archive/14.06/crowds.html).
  15. [www.cambrianhouse.com](http://www.cambrianhouse.com).
  16. [dreamsongs.com/IHE](http://dreamsongs.com/IHE).
  17. [www.technorati.com](http://www.technorati.com).
  18. [www.youtube.com](http://www.youtube.com).
  19. [www.mturk.com](http://www.mturk.com).
  20. [clickworkers.arc.nasa.gov](http://clickworkers.arc.nasa.gov).
  21. [www.globalinnovationjam.com](http://www.globalinnovationjam.com).
  22. [wikipedia.org](http://wikipedia.org).
  23. [www.tropicaldisease.org](http://www.tropicaldisease.org).
  24. [www.p2pfoundation.net](http://www.p2pfoundation.net).
  25. [www.secondlife.com](http://www.secondlife.com).
  26. [money.cnn.com/magazines/business2/peoplewhomatter](http://money.cnn.com/magazines/business2/peoplewhomatter).
  27. [www.chumby.com](http://www.chumby.com).
  28. [www.gartner.com/it/page.jsp?id=495475](http://www.gartner.com/it/page.jsp?id=495475).
  29. [www.redhat.com/about/news/prarchive/2006/earnings\\_2007q3.html](http://www.redhat.com/about/news/prarchive/2006/earnings_2007q3.html).





## 2 Open innovation – the new trend

This chapter presents a few examples of open innovation. The final section will draw some conclusions based on the general tendencies illustrated by these examples. It's evident that open innovation belongs to the current generation. The changes that this generation is initiating respond to further refinements in the notion of innovation. "Disruptive innovation" is the label applied to open source software: an innovation with monumental consequences for the market. Expectations are that open innovation could be equally disruptive in other sectors. The third chapter will reveal the extent to which the software market has been transformed. This chapter will focus on other sectors, which lag a bit behind in terms of innovative force, although it is prominent in a number of cases.

### From consumer to producer

Innovation in products, services and markets has been radically transformed as a result of technological developments. The internet, domain-specific software and lightweight software in the form of webservices have made it possible for users to become involved in competition and innovation. This can even occur without the support of a "conventional" organization. A community can manufacture its own products: like open source software, for example. Expanding the consumer role into one of a (joint) producer – the so-called "prosumer" – helps organizations to improve products, to stimulate R&D and to undertake another type of marketing. This gives rise to the threat that a product made by a community itself might overshadow existing products. The interesting point is that a product need not be completely polished for it to pose such a threat. For this reason, innovation expert Clayton Christensen appropriately labels the raw form of open source software as a "trump card."

### Generation C

Open source innovation, in the sense that consumers actually participate in production, is now something that has completely taken off. Users have also become producers, just as we saw in the context of open source software development. These days, increasingly more people are digitally active. Over one billion people are now online, and this growing population

has been called the “content creating generation”: “Generation C” for short. Half of this billion have already made digital “things” themselves, such as a website or digitally manipulated photos. Companies who know how to reach this digitally productive generation can turn large profits from them. Generation C uses various languages to be creative: from natural spoken and written languages to the artificial languages used in programming. The tendency is for the more complicated languages to become increasingly simple, enabling increased innovation: for example, modular programs or “mash-ups,” which are based on so-called “webservices” may be used in company R&D.

## **Open source innovation**

Together, these factors contribute to an increase in open innovation, along with an increasing prevalence of open sources on which communities can work. New specifications can be added or deleted and, based on these sources, new products can be fashioned that will have their own completely unique role. Just as in open source software, opening product specifications in other fields is an important element for creating innovation. Thanks to rapidly growing opportunities to create divergent products, processes and services based on a digital design, open source innovation is no longer limited to standard software products, such as operating systems, databases, script languages and web browsers. It is, for example, to be found in the CAD/CAM domain, but also in domains related to chemistry and bio-computing, such as the pharmaceutical industry. The trend is that progressively more open-innovation projects will involve domain-specific open source information manipulation.

## **Examples of open innovation**

One can find a rich array of examples of open source phenomena outside of the software industry: a Korean newspaper, advertisements for gym shoes, an internet encyclopedia, a burned-down factory, BzzAgent, a website for pensioners, the “mash-ups” from Amazon, the seeker-solver science market, the open innovation strategy from Procter & Gamble and the developments in bio-computing, genetics, and medicine. These are all examples of the impassioned commitment of individuals and groups that often participate in thought and action for nothing or for a negligible fee. Anyone looking at all these examples from a distance sees a new landscape in which sharing knowledge with the outside world has become common practice and in which consumers and commercial customers actively participate in forming ideas and creating products. Whether the end product is completely digital or is a tangible product such as a chair

or medication makes a major difference in these examples. A second important difference is the language in which people communicate: an internet encyclopedia is written in normal human language, whereas we have to master webservices or CAD/CAM systems in order to be able to participate.

## 2.1 Innovation writ large

Innovation can be modest, consisting of small adjustments that benefit a company in one way or another. Obviously, we would prefer to have radical breakthroughs that knock the competition back on its heels with one heavy blow. Open source software development belongs to this latter type; it is INNOVATION in capital letters. The introductory chapter drew a parallel between Henry Ford and the production evolution that the assembly line initiated. But this parallel doesn't provide a complete view of all that open source innovation entails.

Innovation has a number of variations: product innovation, process innovation, market innovation, organizational innovation and so on. The foundation of this multifaceted concept has been identified by Joseph Schumpeter, one of the most important economists of the previous century. In his *Theory of Economic Development*<sup>1</sup> published in 1934, Schumpeter describes five types of innovation that have a lot to do with the variants that we have just enumerated. For Schumpeter, innovation is as follows:

- the introduction of a new product not yet known to the consumer, or the introduction of a new quality of a familiar product;
- the introduction of a new production method that need not in any way be based on a new scientific discovery;
- access to a new source of raw materials and semi-finished products;
- the re-organization of an industry, such as the establishment or dismantling of a monopoly; and
- the opening of or entry into a new market.

Open source software development is INNOVATION writ large because it manifests itself in all of these five forms. Nevertheless, not everyone agrees that open source software is so innovative. Traditional suppliers upbraid open source communities for displaying mostly copycat behavior and seldom coming up with anything really new. Hence it's not open innovation but, in fact, open imitation. The success of open source in introducing a new product or a new quality of a familiar product, as mentioned at the top of the list above, is therefore subject to debate. But let's just continue down the list and leave this point for later.

The "introduction of a new production method" has already been extensively discussed in the introductory chapter. This type of innovation

is based upon the self-determination and intrinsic motivation of the participants. Additionally, the internet is a necessary condition because it makes distributed work possible. In this respect, Schumpeter's third point, "access to a new source," must be seen as the central theme of open innovation. "Not all the clever people work for us; we must find a way to tap into these other human resources." Access to bright minds outside a company's own organization is, in essence, the greatness of open source. The reorganization of an industry, the fourth type of innovation, was only possible when open source software attained true economic significance. It is also for this reason that we can again stress the innovative power of open source in capital letters.

The final form of innovation (the opening of or entry into a new market) and the first one, which we skipped over for the sake of convenience (the introduction of a new product or a new quality) are discernible in the argument made by Clayton Christensen, an expert on the subject of innovation.

Christensen is a professor at Harvard Business School who continues to build on Schumpeter's ideas. Schumpeter spoke about "creative destruction" when he was referring to large, trailblazing innovations in which old ideas, technologies and skills were made superfluous by the arrival of new business activities. Christensen prefers to use the word "disruptive" to describe trailblazing innovations.

He identifies the open source software movement as a type of disruptive innovation. Remarkably, he considers open source to have a market-disrupting effect on account of "the quality" of its products. In other words, the issue concerning open source about which there is most discussion – the question whether the products themselves are sufficiently innovative – is raised by Christensen precisely to indicate that they "are disruptively innovative." At the Open Source conference in 2004, Christensen explained that disruptive innovations do not always need to be tours de force that contribute something substantial to an initiated developmental path. Disruption can also mean that a break is made from a developmental path in order to investigate a more rudimentary level:

"But then there was this other kind that we call the Disruptive Technology that comes into the market every once in a while, and open source clearly is one of these. And we called it 'disruptive' not because it was a dramatic breakthrough improvement, but, instead of sustaining the trajectory of improvement, it disrupted and redefined it and brought to the market a product that was crummier than those that historically had been available. In fact, it performed so poorly that it couldn't be used by customers in the mainstream. But it brought to the market a simpler and more affordable product that allowed a whole new population of people now to begin owning and using it."

Clayton Christensen, 2004<sup>2</sup>

Clearly for Christensen, the power of open source software is found in its combination of the first and last types of innovation from Schumpeter's definition: the introduction of a "crummier" and simpler product having basic functionality, and (as a direct result) the initially rather tentative opening of a new market. Presently, most open source software is becoming a little easier to use and can also be handled by users who are not programmers. As a result, open source software now competes in terms of price and functionality with software from what Christensen identifies as the traditional trajectory of improvement.

In *The Innovator's Dilemma*, Christensen explains that there are, in essence, two types of disruptive innovations: low-end disruption and new-market disruption. Various types of buyers are associated with each.

### Low-end disruption

Buyers for whom the existing product actually offers too much (Christensen's "overshoot" effect), and who are interested in products that do or can do less than what is being currently offered on the market, represent fertile ground for low-end disruptions.

### New-market disruption

Buyers who could not previously afford the existing products and are suddenly attracted by a new, less expensive offer. Since much open source software has become more user friendly, this disruption will have its full effect.

Clayton Christensen is not the only one who has proclaimed the disruptive character of open source software. The following three quotations also make this point:

#### Open source software development disruptive

"Disruptive technologies like open source software development reduce the margins of existing players, lower the barriers to innovation, and end up expanding the market for players who are able to quickly understand and play by the new rules."

Tim O'Reilly, 1999<sup>3</sup>

"The disruptive effects of the open source production process could be as great or greater outside the information sector, at first simply because of the increased efficiency of information processing that will effect many other economic activities, and second because of the spread of the model of production itself to other sectors of the economy."

Steven Weber, 2004<sup>4</sup>

"The great open source debate [in 2006] will look at how free and open source technology, as a disruptive technology, has changed the way the software industry does business and the new opportunities it has spawned."

*Eweek*, 2006<sup>5</sup>

## 2.2 Value creation with open innovation

Management gurus C.K. Prahalad and Venkatram Ramaswamy outline the urgency of drawing lessons from open source due to the effort that it costs companies to continue to create value. The rapid generation of new ideas is critical and, accordingly, existing ideas become outdated more quickly. Greater involvement of outsiders in the production process, as illustrated by open source software, must become the new rule for marketers and corporate strategists. This is the message that Prahalad and Ramaswamy convey in their book *The Future of Competition*.<sup>6</sup>

In their view, one of the most important reasons to open the production process to customers is the enormous burden placed on the shoulders of managers to create value. It is becoming increasingly difficult to remain ahead of the competition or to hold onto a strongly contended competitive position. In *CIO Magazine*, the two professors make a case for engaging customers in work aimed at creating more value.

"It's time to loop customers into the act."

C.K. Prahalad and Venkatram Ramaswamy, 2004<sup>7</sup>

How do you work in a way that keeps customers engaged in your activities? And is this really anything new? We certainly cannot imagine that large companies (e.g. Philips) ignore customer feedback when undertaking product updates. For sure, it is not as if Philips and others have ignored consumer experiences in the past. On the contrary! But since August 2005, Philips has been using the Leadusers website to conduct more research in this area. This deliberate step toward consumer-led innovations (i.e., user-driven innovation) is a direct consequence of the open source inspired research work by MIT professor Eric von Hippel. At the start of 2005, Von Hippel published his book *Democratizing Innovation*. He is the joint founder of the Lead User Concepts consulting agency, whose clients include 3M, Kellogg, Nestlé, Nortel Networks, and Verizon. The influence of Von Hippel on the (open) innovation practices of these companies is a good example of how an open source best practice can be used productively elsewhere; this is a true open source lesson for innovation that has more than proved its worth, as 3M has reported US\$146 million in in-

creased sales over the last five years thanks to better lead-user focus. The best practice in question, from open source software development, is that the expertise and experience of users is explicitly included. Open source software developers have always been users themselves, and, in part, that remains the case. Now the lead-user approach is shifting to other sectors, as a consequence of Von Hippel's insight, which is rooted in open source development.

The open source software movement developed from an exceptionally strong user-led form of open innovation into a full-grown economy involving large software development organizations as well as newly created companies. Open innovation is moving in the opposite direction. The openness was initiated by strong partnerships among companies and knowledge institutes, which was then followed by collaboration with lead users. Expectations are that the relationship between lead users and companies is going to develop further. Companies would like to make use of the relationship, but they will also likely encounter additional competition in the form of communities and new businesses.

## Wide Open

Just as companies can no longer do without the internet in working out new strategies, collaboration with communities will become common in the future. This is the conclusion of Demos, a prominent think tank in the UK, in its *Wide Open* report.<sup>8</sup> The report has a revealing subtitle: *Open source methods and their future potential*.

In mid-March 2006, Nicholas Donofrio, IBM's executive vice-president of innovation and technology, pinpointed the need for more collaboration in various areas by observing that all the important discoveries had already been made, in the previous century. It is becoming more difficult to come up with something new; a "next big thing" doesn't come along so readily. Consequently, Donofrio identifies this age as one of innovation based on collaboration in services, processes, business models and culture:

"An era of inventions ended. [...] Innovation was a little different in the 20th century. It's not easy (now) to come up with greater and different things. [...] When it comes to innovation, there is a need to think collaboratively and in a multifaceted manner, as this determines who wins and who loses. [...] Innovation today is more about services, process, business models or cultural innovation than just product innovation."

Nicholas Donofrio, 2006<sup>9</sup>



## 2.3 Have others do the work

The nice thing about intimate user involvement is that companies can convert consumer experiences into improved products more directly. Formerly, companies had to use marketing reports and the knowledge of departmental customer contacts to do this; now it is as if users are generating their own solutions directly. The language they use in this process is digital, as manipulation software forms an important link in the chain.

Eric von Hippel, in addition to being a professor and joint founder of Lead User Concept, is also the head of the Innovation and Entrepreneurship Group at the MIT Sloan School of Management. He is concerned with “user-driven innovation” but also with the democratization of innovative processes so that everyone can participate. For more than a quarter of a century, Von Hippel has conducted research into user-driven innovation, but the digitalization of society in recent years has resulted in growing interest in his research. Von Hippel even identifies greater accessibility to software tools, especially in the CAD/CAM domain, as the most important reason why innovation in the future will be able to profit from user ideas.

The notion that user participation can be useful is already quite old. It goes back to a time long before software and other technological aids existed. In fact, Plato wrote in *The Republic* (loosely translated below) that it is precisely seasoned users who have the most experience and must be better involved in production:

“Automatically, we think that the carpenter certainly knows how to make a bed or a table. But is it not so that, in the case of a horse only the rider can actually determine what is the best bit or bridle. It is clearly obvious that seasoned users have the most experience. Users have to tell makers what they want, exactly. In this way, they can be served according to their whims.”

Plato, *The Republic*, 360 BC

Studies in recent years reveal that the relationship between production and user needs is far from settled. In fact, somewhere between 10 and 40 percent of all product users continue to tinker with a product after it has been purchased. Evidently there remain many unfulfilled wants; many products do not satisfy needs. A central case from Von Hippel’s book *Democratizing Innovation*<sup>10</sup> shows how the principle of user-driven innovation works. In particular, the relationship between technology (CAD/CAM and internet) and user needs is interesting. Von Hippel’s case concerns kite surfing.



## Open source lessons for innovation

It wasn't until 1999 that people were seen standing on boards being pulled through the breakers. Yet the kite-surfing market has now grown to more than US\$100 million. A group of kite surfers who were unhappy about the goods available in stores began to make better products themselves. Using the internet, their former site [www.zeroprestige.org](http://www.zeroprestige.org) (the remainder of which still can be found at [www.mit.edu/people/robot/zp/zeroprestige.html](http://www.mit.edu/people/robot/zp/zeroprestige.html)), and modeling software costing US\$195 ([www.rhino3d.com](http://www.rhino3d.com)), they began to develop kites for themselves. At first, these designs had to be completely manufactured by the group, but they soon found companies where their digital designs could be piloted through production. The innovation cycle became a streamlined system and, soon, they were ready to move from design to kite within a week. Their needs could be met thanks to simple IT. The group was satisfied because they had a new and better product that was even cheaper than a kite in the store. The creations were made available on their website, as they did not want to set up a commercial company: clearly an example of open source. The designs were published under a "creative commons license." This is a version of an open source license for creative work other than software programs. When they were contacted by a clever businessman who wanted to bring the kites to market, the group did not have any objections. After all, they had chosen this type of license. However, existing suppliers were immediately confronted with another formidable competitor. The new company had not spent any money on R&D but could still offer a better product at a lower price. For this company, "not invented here" proved to be a blessing.

## Translating user experience into innovation

Von Hippel speaks of "innovation niches" in which users of products play an important role. User experiences can provide suggestions for improvements, and sometimes it is the users themselves who start the implementation process. Von Hippel describes innovation as a process of trial and error: a design is made on the basis of experience (Design), a prototype is made (Run), a test is conducted (Run) and the findings are examined (Analyze).

These steps are found in traditional R&D but also in innovations where users take the lead. The process of trial and error has resulted in innovations for new car parts, coffee machines, bicycles, laptops and websites. Laboratory R&D has the advantage that the tests are run under controlled conditions, making it possible to replicate results. The disadvantage is that the conditions never fully conform to practical reality. In democratic innovation processes (user innovation), we see that users feed their

experiences back into the original product. The closed environment of the laboratory and the open user environment in which innovations take place can be extremely complementary.

## **2.4 Product innovation in the age of coding**

In the type of open innovation that leads to the production of a physical object, collaborative effort on the digital design is fundamental. Thanks to the CAD/CAM technology of Rhino, the kite surfers were ultimately able to convert designs into tangible products. This line of development from digital design to physical end product will be further elaborated in this section. We will begin by considering genetics and pharmaceutical research, areas in which the relation between programming and innovation is becoming increasingly more profound. Next, we will make a foray into simulation and hardware virtualization. Then, after briefly speculating on the distant future of open source in the aerospace industry, we will conclude with a discussion of the Digital Designer software used by Lego.

### **Open innovation is growing**

There are two central developments that make it possible for progressively more innovation to occur outside of organizations. The first one involves an increasingly higher degree of automation. The more the majority of production and products become “virtual,” the easier it is for others to take part. The second development has to do with product consumers adapting the technology. Because the technology is simpler, less expensive and consequently more accessible, the number of people participating in open innovation is on the rise.

### **Open source collaboration in genetics and pharmaceutical research**

In genetics and pharmaceutical research, open source collaboration is beginning to grow in importance. Just as with CAD/CAM systems in which a kite can be developed, the design of a new medicine is increasingly dependent on specific software. The number of web-based and open source tools for bio-computing and gene analysis is steadily rising. For example, the open source platform SourceForge includes AMOS, “A Modular, Open Source Whole Genome Assembler” ([amos.sourceforge.net](http://amos.sourceforge.net)). And prizes are awarded to scientists who support the open source model in bio-computing, such as the Overton prize presented to Ewan Birney by the International Society for Computational Biology.<sup>11</sup>

“Dr Ewan Birney [...] was awarded the 2005 Overton Prize in honor of his advocacy of open source bioinformatics, and his generous contributions to the BioPerl community. Perhaps even more important to biology is his leadership of the Ensemble genome annotation project [...]. Dr Birney [...] credits the success of the Ensemble project to the open source development model.”

Stephen Maurer, a law student at the Goldman Sachs School of Public Policy at the University of California, made an appeal at the 2004 annual biotechnology conference in San Francisco for the application of an open source approach to the development of medicines to treat tropical diseases. The website of the Tropical Disease Initiative (TDI), [www.tropical-disease.org](http://www.tropical-disease.org), provides a means for chemists and biologists to work together. The technology has now progressed to such a point that Maurer expects the greater part of the work to be done without “wet laboratories.” The virtual work can be conducted, for the most part, in an open source manner because the advanced digital tools make this possible. Figure 2.1 illustrates how the worlds of biology and computing are coming together and how this is leading to the production of medicines in virtual pharmacies.

For instance, collective research being done into malaria and other tropical diseases such as schistosomiasis is posted on [thesynapticleap.org](http://thesynapticleap.org). With regard to schistosomiasis, research is being conducted to improve the drug Praziquantel. This open source biomedical study is clearly targeting a number of very specific diseases to which the pharmaceutical industry gives little priority. The Tropical Disease Initiative was conceived to bring about the development and production of medicines for these tropical illnesses. Knowledge of biology, chemistry and biotechnology is brought together in an internet community of universities, laboratories, scientists and institutes. The virtual information manipulation component (computing) in the research plays a key role in this model. Besides the voluntary contribution of researchers, “virtual pharmas” must be established. These are non-profit venture capital organizations promoting the development of promising medicines for the Tropical Disease Initiative.

There are two domains in which open source development in the medical field is viewed as having a great deal of potential. The first involves non-patentable medicines. These are medicines for which the patents have expired and further commercial research is not profitable because protective patents do not exist. Imagine, for example, that aspirin had the potential to cure cancer. The pharmaceutical industry could not be expected to initiate such research. It is more likely that an open source community of scientists and institutes would undertake the task; they might be willing to act for reasons other than monetary gain. The second domain involves medicines that, for some other reason, do not offer any prospect of

commercial success – because they are used in the treatment of very rare diseases, for example.

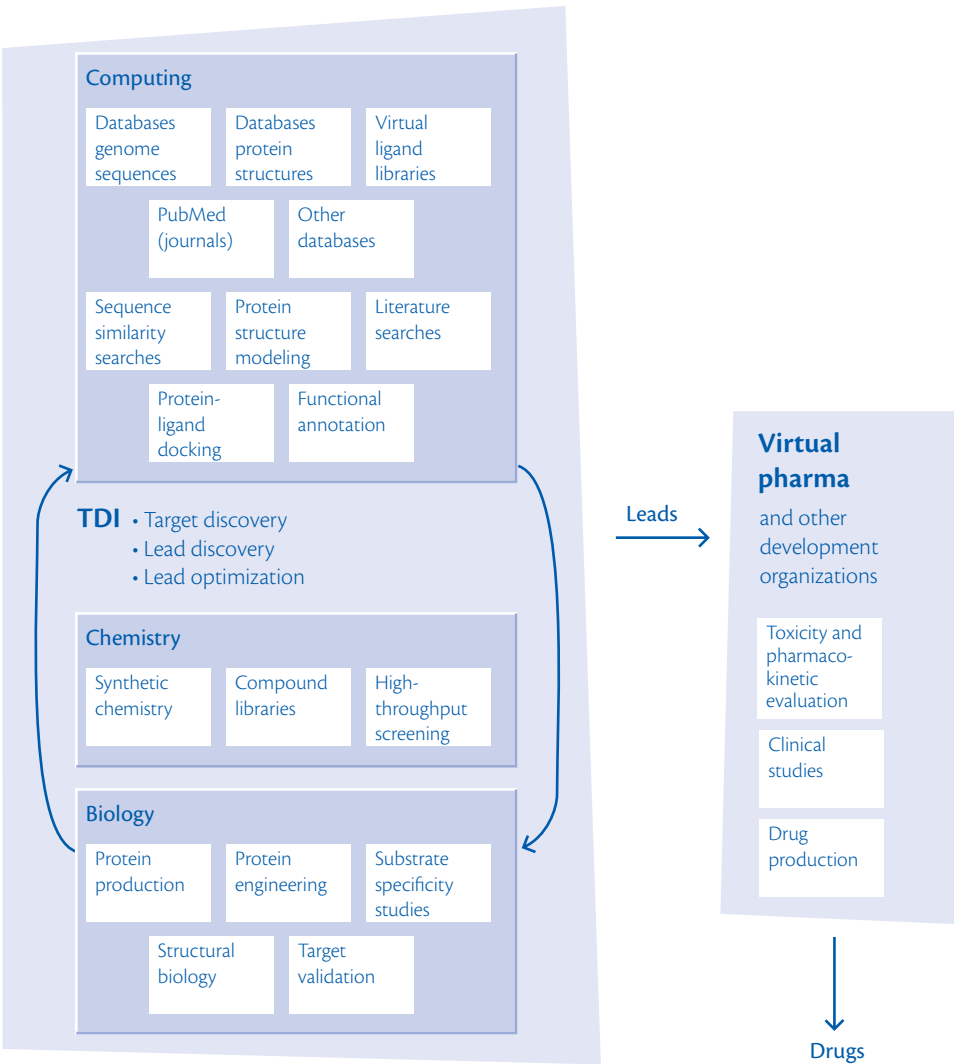


Figure 2.1: Information manipulation (computing) is a key element in the research into tropical diseases

### Factory simulation and hardware virtualization

Factory simulation is a good example of virtual design and testing in a life-like environment. At [www.factory-simulation.com](http://www.factory-simulation.com) it is clear that leading organizations in many different fields benefit from this type of software: Boeing, Braun, DHL, Excel, FedEx, Gillette, Goodyear, Honda, Lock-

heed Martin, Mattel, Nissan, Northrop-Grumman, Siemens, TNT, the US Air Force, the US Army, the US Navy, Volkswagen and Whirlpool. Even the state-of-the-art virtual test lab of Virtutech, which was used in such projects as the Iridium Satellite, fires the imagination in this context.

## **Eliminating virtual deficiencies saves a great deal of time and money**

### **Example: Factory simulation using Flexsim**

Factory simulation can facilitate better management of any given factory. There are a number of ways that costly production time is lost in a factory, ranging from improper allocation of human resources to insufficient supplies of materials. Various components must also integrate seamlessly with each other at the right time. Bottlenecks can be created by one running too far ahead of the other. All producers try to avoid such jams. One way of doing this is by using simulations to provide insight into the operations of a given factory and potentially improve performance. Greater knowledge of the production specifications enables better adjustment of the relevant components in order to avoid problems.<sup>12</sup>

### **Example: The virtual Virtutech test lab**

Nowadays, companies are being confronted by increasing complexity and costs of testing. This demands innovative solutions. One way of resolving this problem involves full-system simulation using virtual hardware, and this undertaken quite early in the developmental process. Recent technological breakthroughs now make it possible for virtualization to construct a software model that can simulate a complete system on a PC. This virtualization is so accurate that it is not only able to program the code directly into the end product but it can connect to other systems that have to work together.<sup>13</sup>

### **Example: Iridium simulates aerospace software with Simics**

Simics has been developed to make it possible for software developers to program virtual hardware so accurately that the software can no longer detect any difference. Each code, from applications to real-time operating systems and device drivers, can run on these simulations. Iridium uses Simics to research how existing and new software works in satellites. It furnishes developers with penetrating insight, as a result of which the performance of flight software can be improved.<sup>14</sup>

## **The production of an open source airplane**

It is an intriguing question whether something like aircraft construction could be completely organized in an open source manner. On the one

hand, the question recognizes the possibilities of open source production and, on the other, exposes its limitations. Charles Gerlach is occupied with this issue. He was a lawyer and a law professor. He worked for IBM as the head of the Global Communication Sector at the Institute for Business Value. Now he operates out of Gerlach Space Systems, of which he is the founder. Gerlach's ideas are rather unorthodox. He would like to get a program off the ground that is aimed at the exploitation of space, including mining such metals as platinum on nearby planets. Gerlach is convinced that open source production can help him realize his goal. His open-source ambitions serve as a mirror for the aircraft construction industry, a high-cost and inefficient sector which barely utilizes knowledge available worldwide.

It costs lots of money to build an aircraft, and it is a risky business. A community is not going to lay down the cash and run the risks. From such a standpoint, the open source production of an aircraft is purely hypothetical. But dividing aircraft production into a physical and a virtual (designing and testing) component makes it a completely different story. A parallel could be drawn with the kite surfing example. Ultimately, the market leader was surprised by a better product from a competitor who had not done any of its own R&D but had used an open source design.

In *Building an Open Source Space Program*,<sup>15</sup> Gerlach explores the limits and obstacles to the application of open source in aircraft construction, which, in Gerlach's view, has fallen behind in adopting open source principles. The supposed lag has been caused by the historical ballast of a technologically driven, hierarchically controlled production process. Open source demands a radical shift. What Gerlach depicts strongly resembles the legacy problems in large software systems. We know that they cannot just become "group sourced." If we want to hand a project over to a distributed community, we would first have to divide up the code into manageable pieces, so that contributors can each work on a portion suited to their expertise.

## **Hurdles to overcome**

Restructuring production is necessary for the success of an open source project distributed over the internet. A second necessary condition is more technical in nature. Whereas a simple CAD/CAM package was sufficient to build a kite in the kite surfing of [www.zeroprestige.org](http://www.zeroprestige.org), aircraft design makes other demands. For an open source production process distributed over the internet to a large number of participants, the interaction between CAD/CAM and Simulation Based Design is essential. Once created, a design can be tested immediately using simulation software. At this time, the step from design to testing is still not fully computerized.

However, the quality of software tools has increased enormously in recent years, and the open source design of an aircraft is now, in principle, within arm's reach. Essentially, we can design and test without much difficulty and without extra investment.

Charles Gerlach imagines his dream realized by the combination of open source production and traditional suppliers. It would be just as in the kite surfing example, but with the difference that testing the designed products becomes purely the work of computers. NASA has already dared to send the Mars Pathfinder on its mission with open source software in critical systems. Perhaps this will smooth the way for a subsequent step: the application of more open source principles in a "space project" such as the one envisaged by Gerlach.

### **Lego uses Digital Designer to open up its innovation process**

Lego has been implementing initiatives to throw open its innovation procedures since 1999, but 2003 was a breakthrough year for the company. That is when Lego came out with "Factory," an internet site on which more than 3,000 visitors have made thousands of Lego designs. Lego selects the best, which then turn up on store shelves along with a photo of the designer on the back of the box. Mark Hansen of Lego has some thoughts about manufacturing products in this way:

"The ideas of what you can do with this is endless. It's just the tip of the iceberg [...] Lego isn't special. Any company out there can do something like this."

Mark Hansen, Lego

Lego Digital Designer is the software for building with Lego and it represents the key to the open-innovation success of the company. This CAD product for 3D designs can be downloaded from the Lego site. Even children as young as seven can participate. Each design is posted on the site and can then be ordered online. The design software automatically delivers packing lists to the factory. The process is depicted graphically in Figure 2.2, although the initial part of Step 2 ("Crates of Lego bricks are imported from Europe") is irrelevant to the European market.

It is custom work for the masses, but it is much more than that. Mark Hansen, the person responsible for Lego Factory, the site where all this is happening, calls it "marketing with consumers" and identifies four essential advantages of this model:

1. feedback from the customer;
2. recommendations by the customer;
3. innovation; and
4. co-creation.





Figure 2.2: Open innovation at Lego thanks to usable design software

Obviously, the direct access to customers is important to Lego. It provides feedback for the company's own manufactured products. This is doubly interesting for Lego, as normally contact only occurs through the parents of the ultimate target group, children. Recommendations by consumers to other consumers have a reinforcing effect. And finally, the creativity of the community can be added to the in-house creativity in co-creation: consumers make their own designs but may also collaborate with Lego's own designers.

People participate under the condition that their design must be original. It must not represent any person or a product of one of Lego's competitors. Once Lego Factory has screened the design, it is immediately posted in the product gallery, along with a quantity of information about the maker such as nickname, age, country of residence, name of the design, and a description. Each submission automatically becomes "public knowledge." Communication with the Lego community occurs in various ways, including through a message board on the Factory site.

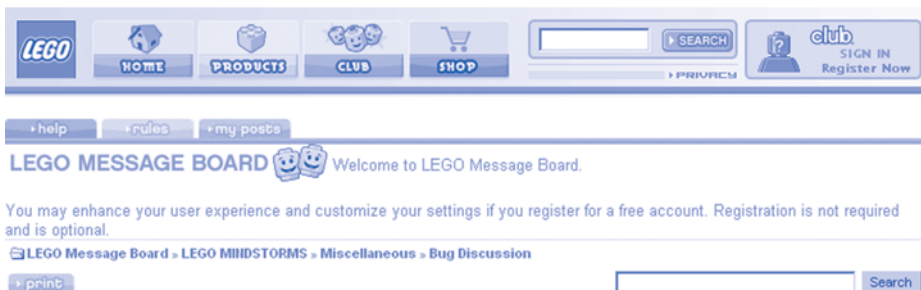


Figure 2.3: Communication among creators occurs on the Lego Message Board



This new form of communication has opened many eyes at Lego; collaboration and co-creation have become permanent elements of the culture. Based on experiences with the community, Lego has formulated ten lessons for open innovation.

#### **Ten open innovation lessons from Lego**

1. Understand how value is perceived.
2. Interact frequently, openly and honestly.
3. Know yourself, and own up to mistakes.
4. Encourage conversations between customers.
5. Seek out the right persons and ask them to participate.
6. Formalize what can be formalized.
7. Participate: you get back what you put in.
8. Don't manage the community but inspire it.
9. Don't sell but show passion and share passion.
10. It's OK to say "I don't know."

This list clearly demonstrates that the sort of "joint" production at Lego means that Lego people must not place themselves above the community. No sales pitches, nothing really to manage, just be open and honest and participate without any pretensions in the community: that is the lesson to be learned.

## **2.5 Open innovation is something typical of our generation**

Children as young as seven are working with the CAD package from Lego. How long will it be before they begin to participate in other open innovation programs? Increasingly more people are becoming digitally active and more adept in using the computer. They make open source software, work with CAD/CAM and/or communicate with companies over the internet using plain language.

You must be quite skilled in order to take part in designing an aircraft, but a kite surfing design is not easy either. The more that software improves and is simplified, the more these things come within reach of an increasingly large group. Development is handled by individuals capable of dealing with the digital processing of information. Our generation is characterized by individuals who are becoming progressively digitally active. At the beginning of 2004, Trendwatching.com, an international network of 8,000 trend analysts, coined the name "Generation C" for this group. The "C" stands mainly for content, creativity and community. "C" for "consumer" should not be included, since what makes Generation C unique is the fact that it is concerned with production rather than consumption. There is a shift from consumer to "prosumer": the co-producing consumer enabled by modern technology and the new media.

The trendwatch agency outlines a number of characteristics of the new generation:

**Generation Content...**

is creative; creativity is the path to content. We are all creative. Being creative is also permissible; parents are no longer concerned to limit the creative aspirations of their children, as it is now evident that such careers have practical potential. Generation C wants to be the “master of their own destiny,” or at least have the idea that they are in control and not governed by others. They want to be praised and honored; they want to be famous.

The music vendor TMF makes optimal use of the Generation C trend by having its viewers co-build personal profiles on the TMF site. Other media are also setting viewers and listeners to work. MTV has its *Starzine*, an online magazine that is fully produced by viewers. They can send photos taken with their cell phones to *Starzine* and have them appear in their own magazine. “Snap, Send, Shine. Fame is just a click away.”

But even old-school broadcasters like KRO (the Catholic Broadcasting Station in the Netherlands) have now discovered that they can use audience creativity. With “build your own altar” KRO has bridged the gap between their established identity and engaging the new generation to create for themselves.

Generation C is active on internet blogs. At [www.lulu.com](http://www.lulu.com), young people are able to realize the *Starzine* credo: “fame is just a click away.” The online publisher offers the public the opportunity to publish books on the internet and to print them in any desired format and layout, on demand. Even the price can be set by the customer. Notably, Julie Powell has now sold more than 100,000 copies of her book, and in 2006 she won the first Lulu Blooker prize, the prize for the best blog book. Lulu.com presently numbers more than 40,000 books and grows by 10 percent every month. It is no coincidence that the founder and boss of Lulu is also the co-founder of the Red Hat open source company.

Below we present two examples of open source production in the clothing industry. It’s strikingly evident, even in these examples, that a competitive element is often employed to mobilize the creative class. At Threadless, we can win prizes, become famous and make friends. In the case of Fluevog, money is not involved at all; instead, it’s all about recognition, which is what John Fluevog believes is at the heart of open source.

## Threadless T-shirts

“Win fame, friends & \$500 in cash & prizes” is the teaser for Threadless, an online-T-shirt design competition. Anyone can participate and send in his or her own T-shirt design. The Threadless community chooses a winner once a week. The shirt is printed and can be ordered. Threadless is a manufacturer that makes clever use of Generation C. There have already been 30,000 submissions to date and 1,500 more are added every week.



Figure 2.4: Become famous, design a T-shirt ([www.threadless.com](http://www.threadless.com))

Threadless has now built up a collection of two hundred different T-shirts.

## The open source shoes of John Fluevog

“Open Source Footware”<sup>16</sup> is an initiative of shoe designer John Fluevog. This successful Canadian has stores in the United States and Canada. As part of his own open source project, he invites customers to help co-design the new collection. This represents enough of a promise of eternal fame that people willingly participate, as they might be rewarded by becoming a co-designer of an original Fluevog shoe.

- The shoes are named after the designer.
- The designer is made a member of the Fluevog Design Alumni, a select group of people whose designs have been chosen.
- The name of the designer appears on the box and sometimes even on the shoes that are actually introduced into production.

You've Heard Of Open Source Software - Get Ready For **JOHN FLUEVOG'S**  
**OPEN SOURCE FOOTWEAR™**  
 "But how can a software concept work with shoes?", we hear you gasp in astonishment. Keep reading.

**YOU ALL KNOW** how Open Source software works, right?  
*Sure you do:* everybody has access to the program's source code and anybody who thinks of an improvement can just write it up and send it in. If other users like it, Yahtzee! Everybody wins! It's simple, it's people - driven, it's non - monetary and it makes software that people really want. *We're going to do shoes the same way.*

It's a concept that's, well, *maniacally tremendous:*  
 Got an idea for a shoe? Even just for part of a shoe?  
 Scribble it down and send it to me. I don't care if it's  
 on a bar napkin, as long as I can make it out.

**ARE YOU FRUSTRATED**, not finding the shoes you really want? Is your imagination ahead of the whole shoe industry and you're sick of waiting for them to catch up? Here's your chance to go over their heads and deal with someone who actually cares what you want. All you need is that brilliant idea. Fax it, mail it, upload it, email it, bring it in - just get it to me!




Figure 2.5: Purchase open source shoes or design them yourself

The example of open source footwear might seem somewhat frivolous, not in the least because of the way in which Fluevog presents the whole idea. With humor and equivocation touching on irony, he summons everyone to take part. Hi-tech equipment is not necessary; a scribble on a beer mat might be enough. John Fluevog regularly goes through the latest submissions when he is looking for new designs. After two years, more than three hundred serious submissions have come in and ten shoe designs have been selected (and named), such as the "Anastasia" shoe model, sent in by a woman in Russia.

Despite the sometimes comical character of these examples, the impact of such developments should not be underestimated. The macro-economic importance of Generation C was emphasized in 2002 by Professor Richard Florida in his book *The Rise of the Creative Class*.<sup>17</sup> Creation by individuals is the current trend, and companies able to draw on these new sources of creativity are cleverly exploiting them to their advantage.

The entire content-creating generation is naturally much larger than the community of software developers. More than half of the people who are online have made their own digital content,<sup>18</sup> ranging from photos to poetry, videos, and live webcam images. If we apply American statistics to the entire online population, we would estimate that there are half a billion individuals who belong to Generation C.

A substantial proportion of the world population can be regarded as digital producers. The only difference between amateur and professional is that the pros are paid for their work. Canon recently emphasized this fact in the advertisement for its Optura camera by accompanying the photo in Figure 2.6 with the caption, "It leaves one difference between you and a professional. They get paid."



Figure 2.6: The difference between the amateur and the pro: the pros are paid for it

## 2.6 Webservices: light-weight software for business innovation

One factor accelerating the development of open innovation is the emergence of webservices. With a webservice (a modular piece of software), a given self-developed service can simply be linked to other services. Conversely, the webservices of others can be added to the services you provide. Webservices offer various options that promote open innovation:

- obtaining external knowledge using new services developed by others as a sort of Lego block to be added to your own services;
- allowing outside people to participate in the development of webservices for your own organization in the same way that people co-build open source software;
- allowing others from outside your own organization to continue to build up their own developed services, producing new organizations as frequent and direct spin-offs of the original.

Webservices are popular. It only takes a few engineers to make something interesting and appealing, therefore it becomes much simpler to program the business, as it were. The *Financial Times* used the word “disruptive” often in describing this development in its issue of mid-November 2005. Learning from the experiences of the first users and then making adjustments contributes to success. Let that also be a typical open source characteristic.

### **We no longer have to do much to produce new business applications**

"[Bill Gates and Ray Ozzie warn] that the latest phase of online innovation [is] likely to be "very disruptive" to the industry's established powers. "This next generation of the internet is being shaped by its grassroots adoption and popularization model [...]"

"Tremendous software-and-services activity is occurring within startups and at the grassroots level [...]"

"To those who are in the midst of it, it amounts to an entirely new way of producing and delivering software [...]"

"The Web 2.0 crowd has discovered how to create internet services with mass market appeal on a shoestring. The watchwords of this approach: wherever possible, develop 'lightweight' software from standard technology building blocks that can be released quickly over the web, then learn from the experience of early users to refine the service [...]"

"With Web 2.0 you can be talking about just a couple of engineers building something interesting and compelling."

*Financial Times*, 17 November 2005, p. 15<sup>19</sup>

We are increasingly able to bundle the appropriate software components together in order to build powerful and compelling internet applications swiftly. The webservice approaches of Amazon and Google are particularly good examples of this practice. Accordingly, the IDC research bureau made the following predictions about development in the IT sector in 2006. Five central ingredients are brought together in IDC's forecast: disruption, open innovation, open source, communities and the modern internet (the Google effect or Web 2.0).<sup>20</sup> As IDC puts it:

### **IDC on the IT sector in 2006**

We observe a critically new ingredient and that is the acceleration of disruptive business models – opening innovation – in IT products and service development (the open source effect) and the online delivery of IT in the form of services (the Google effect).

The "I-go-forward-completely-on-my-own" model of innovation has become an extinct species in the IT industry and the incorporation of a community-based innovation model (such as open source) is very rapidly becoming an important ingredient for market leaderships. IDC is convinced that the building up of open-innovation communities will be the big focus of IT leaders in 2006, including Microsoft.

IDC distinguishes the Google effect and the open source effect. The Google effect is associated with the accelerated development of new disruptive business models. The IT sector and business overlap each other at this point, as Google is a digital company and a business service provider at



the same time. The open source effect is related by IDC to communities, and IDC astutely remarks that companies without communities are a dying breed.

### **Webservices compete with open source developers**

The open innovation strategy of eBay is, to a significant extent, concerned with community involvement in the building of eBay webservices. eBay appeals to individual developers and therefore competes using open source projects. Jeffrey McManus, who was responsible for the webservice program at eBay until April 2005, explains on his website how he was apprehensive about a forum discussion organized by publisher Tim O'Reilly.<sup>21</sup> The discussion was to be about the threat that webservices posed to the open source community. It is true that the webservices of eBay are less free and open than open source software. But a software developer can apply the functionality in a way similar to how open source software is used. The big difference involves the conditions under which webservices may be used. Instead of the critical questions about the ethics and freedom of his webservices that McManus had expected, a typically lively discussion took place concerning the possibilities of getting involved and participating.

Evidently, developers still have a long way to go in embracing creativity. In Chapter 4, we will delve more deeply into what motivates people to join these communities.

## **2.7 Open innovation at Amazon, Google and eBay**

The internet icons Google, Yahoo, Amazon and eBay have already found a way to incorporate open innovation. Organizations and developers can make use of the search engines, webstore and the world's largest auction for their own purposes. We in IT are accustomed to speaking about re-use, but in this context the talk is about "mash-ups." This is a term from the hiphop scene meaning a mixing together of two or more soundtracks. Bret Taylor, product manager at Google, has great expectations of comparable mash-ups, but this time in the areas of software and functionality.

"Frankly we like new and innovative solutions. We expect new and creative ideas to come out of this that we haven't thought of yet."

Bret Taylor, Google product manager, 2005<sup>22</sup>

"The web was originally designed to be mashed up," says Google developer Aaron Boodman in *BusinessWeek*.<sup>23</sup> Boodman is the designer of Grease-

monkey, a program for mash ups. It is used, for example, on the Amazon website to display the prices of competitors, as shown in Figure 2.7.

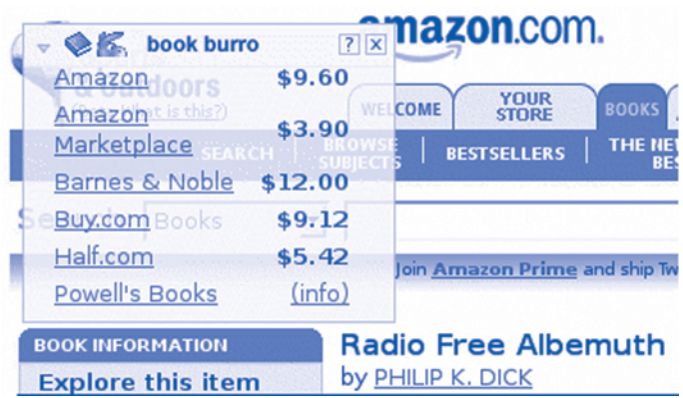


Figure 2.7: Competitor prices on the Amazon site as a result of a “mash-up”

In this way, information can be mixed and new services created based on re-use. The notable thing is that this picture makes it seem as if Amazon were providing the information itself. This is how it works:

“Greasemonkey lets you mash-up websites. It lets you extend and script websites and integrate that script right into the original site as if the designers had intended it to be there. It lets you use their web site, their data, their servers, their work to serve your purpose and function. There will soon be an army of hackers enhancing every site you use. Whether that site likes it or not.”  
NIVI blog, 2005<sup>24</sup>

In mid-2005, Google released the equipment needed to make mash-ups based on the map information of Google Maps. A little later, its competitor Yahoo made its own maps freely available. The official release of the Application Program Interfaces (APIs) formalized what was already happening, as many developers were making use of the maps. Google and Yahoo now want to allow their map data to be mixed with sites in order to perform tasks such as finding personnel or recommending prices. Maps with zoom-in capability can now be found on many sites thanks to the mash-up opportunities allowed by Yahoo and Google. “Wineries in the area” or “houses for sale in my price class” are subjects that mash-ups can address.

An interesting mash-up can be found at [www.chicagocrime.org](http://www.chicagocrime.org). It can be used to investigate the extent of criminal activity on each street in the area around Chicago. This is information that the local tourist bureau or a realtor cannot provide but may have a great influence on the search for a new home.





## Amazon's Mechanical Turk

Amazon's Mechanical Turk is a humorous example of open innovation. It's so funny precisely because it showcases innovation solving the question of how to get all the bright minds throughout the world working for you. This example and the following (NASA's Clickworks) show that web services can be used to outsource standard work to people outside an organization.

In November 2005, Amazon announced a surprising new beta application involving an utterly simple task: selecting a good photo from a set of about six comparable images. There are companies whose services involve photographing every building on every street in a city. From the many thousands of photos made each day, a selection must be made of those that provide the best images of the building. This task quickly becomes labor intensive. The internet provides a way out, as the work can be outsourced to anyone who is willing to do it.

The special feature of this application is that anyone who takes part is paid. A few cents for each selection. Whoever works hard continuously can easily earn a few dollars an hour. The money is credited to a personal Amazon account and, if necessary, transferred to a person's bank account. Amazon names this service "artificial artificial intelligence" with an ironic reference to the artificial intelligence that we're familiar with. Instead of computers imitating human intelligence, a web application taps into the intelligence of thousands of people.



Figure 2.9: Amazon's artificial artificial intelligence

Amazon calls this, appropriately, its "Mechanical Turk." The term refers to an apparatus that Wolfgang von Kempelen developed in 1769. It was a chess machine that could beat everyone and looked like a human, a Turk in this case. The hidden reality was that the actions of the machine were directly controlled by a man in the background. Amazon's "artificial" artificial intelligence works in the same way. You expect that the computer does the work, but ultimately it is performed by people in the background.

## Clickworks from NASA

NASA also outsources work over the internet in a similar manner. Recognizing craters on Mars, to determine the age of areas on the planet, had

previously been the work of highly paid scientists. Through the internet, this can now be performed by anyone who wants to circle them.<sup>26</sup> Again, this is monotonous work that, in accordance with the principle of Amazon's mechanical Turk, has been turned over to a large group of volunteers.

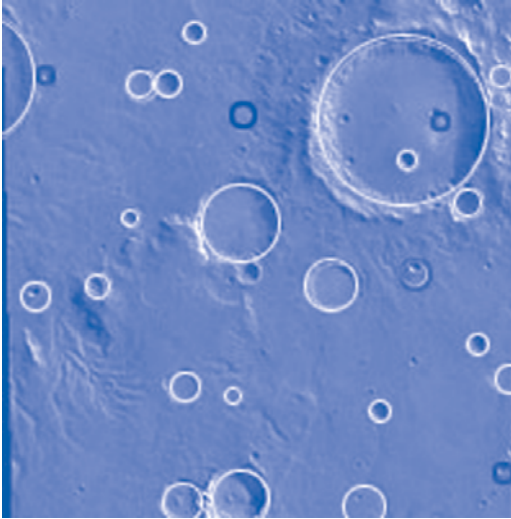


Figure 2.10: Clickworks: NASA's Mechanical Turk

## 2.8 Open innovation at Toyota

In April 2006, *BusinessWeek* published a large-scale study of 1,500 international companies.<sup>27</sup> The study examined the innovative power of organizations and ranked the world's top most innovative companies. Toyota placed fourth. They earned this ranking not only due to a very innovative product, the hybrid car, but also because of their process innovation. Toyota is identified as "a master of manufacturing innovation."

Toyota is intensely concerned with maximizing the use of knowledge inside its own company walls. Principles of self-determination, such as those prominently encountered in the production of open source software, are employed by Toyota to make the greatest possible use of in-house knowledge. In this respect, Toyota reflects a larger if somewhat tentative trend. Many companies experimented a great deal with self-managing teams in the 1990s, although none of them went as far as the self-management found in open source software production. But Toyota's practices are interesting because the company was forced by circumstance to put the principles of self-determination into more extensive practice, and so discovered that they function remarkably well. Admittedly, this extension of the principle is not directly related to self-determination inside the company, as it involved partners and suppliers. All the same,

the success in this case has strengthened Toyota's conviction that this organizational principle actually functions well, and has boosted Toyota's confidence in applying it internally.

Toyota now makes use of a number of open source organizational principles. In the production lines, where employees enjoy a large degree of self-management, and in a crisis situation, Toyota has discovered that an open source type of production can be extremely effective.

Philip Evans and Bob Wolfe of the Boston Consulting Group see little difference between how Linux was created and the manner in which Toyota operates. The lesson that Toyota learned after the disaster of 1997 has turned this company into a hybrid organization with a combination of self-management features and top-down management practices. This makes Toyota one of the first industrial companies to apply the laws underlying the new economic order in the IT sector to industrial chains of production.

## The crisis

On February 1, 1997, fire broke out in the factory of supplier Aisin Seika. Within a few hours, nearly all the machines and inventory went up in flames. Aisin Seika manufactures car parts, the so-called P-valves that control the supply of brake fluid. But even more importantly, it supplied 99 percent of Toyota's P-valves. When the fire broke out, it caused a major problem for Toyota. Because of its just-in-time supply system, Toyota quickly ran out of parts and car production ground to a halt.

In a "rescue-Toyota-from-the-fire operation," it was decided that Aisin Seika's designs for car parts would be given to any suppliers who asked for them. Suppliers would also be provided with all the associated company knowledge (in brief, the complete "source code"), including a small amount of production equipment salvaged from the fire. Production was started in more than sixty locations; participants in the rescue operation ranged from Toyota's own factory to suppliers from outside the sector, even a sewing-machine manufacturer. After three days, a supplier of welding equipment was able to deliver the first thousand parts to Toyota. Other suppliers quickly followed. This way of working possesses the following open source characteristics:

- There was a great deal of improvisation.
- There was no command and control structure.
- A great deal of effort was expended.
- Everyone worked together.
- No one was paid for his or her efforts – one month after the disaster, Aisin reimbursed the direct production costs of those of its rescuers who were in need.

On February 3rd, production had been completely halted, but two days later it was again up and running.

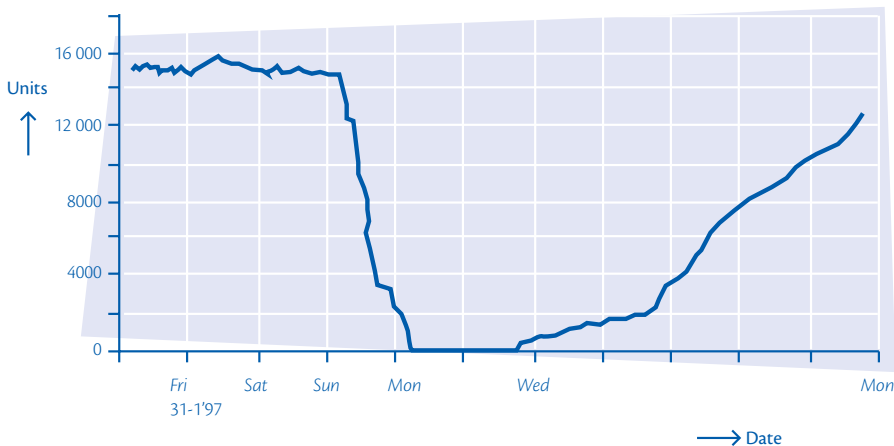


Figure 2.11: Production back in operation as a result of an open source approach

Evans and Wolfe make the comparison with a serious attack by hackers on a Linux server in Italy in December 2003. At the University of Trieste, an attack on a Linux server seriously damaged part of the kernel programming and caused a second problem in the file exchange mechanism. The Italian system manager called American and Australian members of the open source community. Within a few hours, the leak was found and a day later a patch was ready to be sent throughout the worldwide Linux user community. A few days later, the system managers set up a trap to trace the hacker. Twenty people participated in the entire operation and, after a successful conclusion, everyone was thanked and acknowledged by name. This case involved the same characteristic manner of collaboration that had occurred in the Toyota crisis.

**Lower transaction costs**

In a crisis situation, Toyota embraced unorthodox management: no contracts, no instructions, no hierarchy. Instead, there was trust that everyone would do their best to find a solution as quickly as possible. Without wasting time and money on countless meetings, action plans, project goals and extra management, Toyota found a very inexpensive and efficient means to re-establish production. Typical Japanese discipline, is what skeptics might say. But yet....

The current Toyota organization has not adopted the open source approach in its entirety. For instance, there is certainly a traditional hierarchy and there are conventional labor agreements with employees, who are

paid for what they do. Still, Toyota presently allows more self-determination than other manufacturing companies. Toyota personnel management does not focus on personal goals. Company pride and peer esteem are valued highly. Employees are given the latitude to experiment on the production line when they have ideas about improvements. Toyota capitalizes on the expertise and the intrinsic motivation of its employees. Similar to the ways in which open source projects use simple tools as listservs and e-mail, Toyota prefers to make work-floor operations as simple as possible. Tape marks on the floor indicate the time designated for activities, and production can be followed on monitors. Everyone can view the information: the head of production in Canada knows precisely what the hitches are in Japan.

The Boston Consulting Group advises companies wanting to implement open source techniques to build thriving human networks.<sup>28</sup> To achieve this requires simple technology. Keep it simple and open. Simple tools are the basis of good collaboration. The advantage of choosing simple, standard technology is that it creates simple semantics and work procedures. In addition, the work must be as transparent as possible. If there is no good reason for hoarding information, then release it. Allow people to filter their own information and extract what they require. Modular thinking helps, because it squares with Business Process Re-engineering, a theme that has been resounding through companies since the last decade of the previous century. Modular thinking and working is a standard ingredient in open source projects. And teamwork is the decisive fourth ingredient: reward the group, and the group will reward you. Celebrate the successes of a team and dismantle individual performance bonuses. There are four ingredients of successful human networks:

1. Choose simple technology as your standard.
2. Make work transparent.
3. Think modularly.
4. Encourage teamwork.

## 2.9 Open innovation at Procter & Gamble

Procter & Gamble has identified open innovation as a strategic goal and decreed that half of all innovations must come from outside the company. A two-track strategy is pursued to achieve this goal: first, there must be active management of intellectual property, such as Henry Chesbrough advocates in his book *Open Innovation*,<sup>29</sup> and second, there must be involvement of innovation communities outside the company's own organization.

The recent success of innovations at Procter & Gamble was attributable to a rigorous turnaround in the manner in which innovation is conceived. “‘Not invented here’ was somewhat a discovery of Procter & Gamble” jokes

Jeffrey Weedman, vice-president of external business development at Procter & Gamble. Now, he says, “We need to be very open to the idea of bringing in technologies, businesses, and products from the outside.”<sup>30</sup>

The not-invented-here syndrome has been allowed to run its course, and now open innovation has become a part of company strategy. It is increasingly more difficult for Procter & Gamble to generate enough big ideas to keep the engine of a large company running. In 1999, the new CEO at the time, Durk Jager, gave enormous impetus to the company’s own internal innovation. This appeared to be the solution, but it soon became clear that doing this during a re-organization was asking too much. In 2000, Alan Lafley took up the baton and immediately announced the goal of having half of future innovation come from outside Procter & Gamble. According to the company’s own estimates, one and a half million scientists from all over the world possessed expertise that was relevant to Procter & Gamble. In setting the new course, the management of intellectual property came under scrutiny. Previously, Procter & Gamble had only traded intellectual property with which it could do nothing itself; now licenses were being issued regarding all patents. Three years after Procter & Gamble has brought a product to market, or five years after it has been granted a patent, any other party can obtain a license for that product. This gives Procter & Gamble the lead as the first on the market and, at the same time, generates innovations over the course of time as well as licensing revenue. Additionally, the new licensing policy compels other companies to compete with Procter & Gamble. Knowing that the patent is going to come on the market, you would have to think twice about developing something similar.

## 2.10 Open innovation in seeker-solver networks

In the hunt for knowledge outside of the organization, Procter & Gamble and such companies as Eli Lilly make eager use of knowledge brokers such as Innocentive and NineSigma. These companies present their problems to networks and acquire new ideas and solutions from them.

Innocentive is a spin-off established by its parent company, Eli Lilly, in 2001. The organization has since built up a community of 75,000 problem solvers, spread over 175 countries. Innocentive gathers problems from the laboratories of companies such as Dow, BASF and Procter & Gamble and posts them on the internet. The solution that satisfies the criteria for success and best solves the problem is awarded a monetary prize. One of the people who have collected this award is Werner Muller, the former head of R&D at Hoechst Celanese. After retiring, Muller built a small laboratory in his house. He was awarded US\$25,000 from Innocentive for his solution enabling molecules to be produced in a less expensive manner.<sup>31</sup>



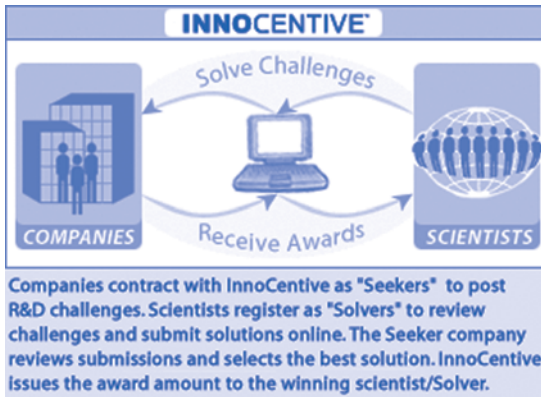


Figure 2.12: Scientists help companies and gain an opportunity to win an award

Research into the economic impact of Innocentive indicates that it is certainly worthwhile to innovate in this manner. From an investment of US\$400,000, income and savings have amounted to more than US\$10 million.

A similar community with which Procter & Gamble has a strategic alliance is the one developed by NineSigma. Here, the solution providers consist of organizations rather than persons. Volvo, the University of California, Dow, Arthur D. Little and Siemens belong to this group. Clients of NineSigma include Procter & Gamble, Kraft, Dupont and Unilever.



Figure 2.13: Innovation managers and solution providers meet at NineSigma

This open innovation model operates in a reasonably closed environment. Requests may be posted anonymously and, if an innovation is developed in response, contractual agreements about intellectual property can be formed.

Anyone seeking to tap into the knowledge of retirees can call on YourEncore. Procter & Gamble and Eli Lilly are the network's founders.



Other companies, such as Boeing, have also become members. The business model of YourEncore is different from those of the above two networks. After a matchmaking process, the employees are hired on a temporary basis. A bonus of 20 percent of the retired person's salary in his or her final year at work goes to the network organization. At the start of the assignment, employees sign a contract that all intellectual property belongs to the organization for which they're going to work. This arrangement is not as casual as an open source project (although Linux kernel programmers are, for that matter, also regularly employed by large IT companies). However, it is certainly an attractive option for companies interested in offering their own labs to provide a second career for the competition's top researchers.

## 2.11 Open innovation in the press

Producing a newspaper collectively has become serious business since the South Korean *OhmyNews* was launched in February 2000. Still more interesting than its rapidly growing readership is the role that *OhmyNews* has played in the recent elections. The victory of President Roh Moo Hyun is largely attributable to the efforts of this open source newspaper. The president himself was the first to acknowledge this fact and, after his victory, thanked the many volunteers working in this new medium.

When an article is posted in the online newspaper, the writer receives a modest fee: a 750-word piece given the heading "Top News" pays US\$11. Evidently, an *OhmyNews* correspondent is not primarily working for the money. All articles are immediately published, but the heart of the newspaper consists of articles from the newspaper's own professional editors, supplemented by vetted articles from the community.

At this time, *OhmyNews* numbers around 38,000 "correspondents" and is read daily by a million people. A printed edition is published weekly. An English-language version of *OhmyNews*<sup>32</sup> has also been recently launched, originally with three hundred correspondents, a number that will likely grow to ten thousand within the first year (2006). Editions for Japan and China are expected to follow.

Active participation in the newspaper instead of passive reading is what distinguishes *OhmyNews* from the traditional press. In an interview with CNN,<sup>33</sup> founder Oh commented on this feature:

"Our slogan is 'every citizen is a reporter.' We've created a new kind of journalism. We call it 21st-century journalism, two-way journalism. So the readers are no longer passive. They are very active and participate in what they want to say."  
Oh Yeon Ho, 2005



Figure 2.14: OhmyNews, the newspaper that helped the new president into office

OhmyNews earns money, but that is not the most important point, according to Oh. “I have to make money,” he says, “but I am not an expert at that. Deep in my heart, I am still a reporter.”<sup>34</sup> It is important that the newspaper remains true to its ideals. The majority of its revenue comes from advertisements, and 20 percent comes from selling articles to other news sites. Earnings from advertisers must not, Oh says, become too large.<sup>35</sup> Other income sources will have to grow: possibly including voluntary contributions from readers to reward a particular article, which is a concept that we saw earlier with Threadless T-shirt designs.

## 2.12 Open innovation in advertising and marketing

Advertising and marketing are two additional areas making increasingly frequent use of open innovation. This not only results in greater creativity but also a stronger bond with users and specific target groups. Below, we will provide a number of examples of open innovation in advertising and marketing. The next section will present the case of open source beer. It will show that open innovation, advertising and marketing are all interlinked. Therefore we can explicitly undertake specific advertising and marketing activities “in an open manner.” But in the case of open source beer and also in a large number of other cases, open innovation necessarily means taking a different approach to marketing and advertising.

Advertising agencies use terms such as “viral marketing,” “buzz marketing,” and “word-of-mouth advertising” for campaigns that do not use traditional media but are directed toward peer-to-peer influence. Customers and consumers play an important role in this by, for example, designing advertisements or by communicating in customer networks and, as a result, convincing others. James Cherkoff, a consultant for collaborative marketing, has made an insightful examination of what this entails. In his open source manifesto, he makes eight recommendations for revitalizing marketing and advertising:<sup>36</sup>

1. Give customers direct access to their own “brand source” in order to convert them into collaborating producers, just as Linux programmers have access to the source code.
2. Spur the fans on and have them to do the majority of the work.
3. Be a brand host.
4. Listen to what the community has to say. Open source marketing means hearing all the rumblings in the market immediately.
5. Get real. Authenticity is one of the most important values in the transparent world.
6. Accept the fact that your customers are cleverer than you are.
7. Let go. Turn ownership of your brand over to the people.
8. Be open minded about these new developments.

### **Involving people in commercials**

Audi, Nike, Mercedes, General Motors and many other organizations involve their customers in the creation of their commercials. They surrender part of the control over their corporate communications. Co-creation means involvement. A commercial from Audi in which consumers were allowed to take control, sketch their own plot and even play the role of a character in the story attracted half a million participants. Tasks, photos and videos were distributed on various sites, such as [www.stolena3.com](http://www.stolena3.com). The participants could submit their own findings, and many began websites and blogs. The campaign was a great success. It produced more than 10,000 leads, an enormous increase in visits to the Audi website and 3,500 test drives.<sup>37</sup> Based on Audi’s measurement criteria, this campaign was nearly 80 percent more effective than other campaigns.<sup>38</sup>

Nike also has allowed consumers to take part in the creation of commercials. Customers were asked to make a thirty second video clip for the Converse brand. Anyone with an idea and a camera could participate. The first seven films were posted on the internet, which attracted still others.

Sales of Converse sneakers rose 12 percent in this period, and the website attracted 400,000 more visitors a month. MTV broadcast the winning commercials. The filmmakers received US\$10,000, a pittance compared to

a “real” commercial but, at the same time, a substantial sum for an amateur. Additionally, there was a great commotion about this campaign, and it produced a lot of “buzz.”



Figure 2.15: Amateur videos for Express Yourself at Converse<sup>39</sup>

General Motors did the same. GM asked consumers to make a commercial, which produced 2,600 submissions as well as three times the usual traffic on their website.

In the Netherlands, consumers supplied the script for a new advertisement for Centraal Beheer, an insurance company. Under the motto “Win your own *Gouden Loeki*,” the Dutch television prize for commercials, Centraal Beheer launched an appeal and received eight hundred ideas for a commercial. Amsterdam resident Frans van Gerwen won. The commercial was, in fact, submitted for a real *Gouden Loeki*, but was not awarded one of the prizes.<sup>40</sup>

**BzzAgents**

More than two hundred thousand Americans and Canadians can call themselves BzzAgents. They participate in a network of volunteers who enjoy making advertising for a specific brand: a “community of communicators.” BzzAgents do get paid, but they also get perks and can earn points. Sometimes participants receive products for testing before they are introduced to the market, and they may have direct contact with the companies that make these products.

Founder Dave Balter has patented his BzzAgent system in order to get word-of-mouth advertising off the ground. Companies wanting to know or influence what customers think and say about their products can call on BzzAgent. “Buzz agents” are then selected on the bases that they belong to the target groups and can be evangelists for the product. This amounts to a further metamorphosis of user-driven innovation that Eric von Hippel proposes in *Democratizing Innovation*. The result can be a word-of-mouth campaign, but it can also serve as a feedback mechanism for customer responses or as the basis of a “normal” ad campaign.

One of the special projects in which BzzAgent was deployed was the promotion of Creative Commons, an open source license for texts, books and music. However, BzzAgent also works for ordinary products, such as beer, books and cornflakes. Organizations like Penguin Books, Anheuser Busch, DuPont, Sun Microsystems, Nestlé and IBM have used the “buzz services” and praise their quality.



Figure 2.16: Whoever wants to approach a community can call on BzzAgent

### Jones Soda

Jones Soda, a soft-drink producer with US\$27 million in sales, has customers design the appearance of the bottles. Anyone can send in photos, and, on the basis of a vote held on the site, a weekly winner is chosen. You might see your own vacation snapshot on your favorite drink or find it on supermarket shelves throughout the country. Many photos have been used, so it is difficult to send every winner his or her own bottle. It is easier, especially for Jones, to have you order twelve of your bottles on [www.myjones.com](http://www.myjones.com) for US\$48.95, including shipping. A pat on the back for your voluntary efforts is not to be expected from Jones, and even sending you a free bottle is too much trouble. Nevertheless, this does not hamper either consumers’ creativity or product sales.

## 2.13 Open source beer from Brewtopia

There are various initiatives focusing on open source beer, but Brewtopia is certainly one of the most interesting. Obviously, anyone can release a recipe for beer so that everyone could make the same beer. Beer recipes

are nothing special. Only a few ingredients are involved: water, hops and malt. However, Brewtopia has done more to capitalize on the concept of open source. They have unleashed the idea on the whole organization. Not only is the beer open source, the entire management from the ground up is based on a number of open source principles. Where the idea came from is a matter of conjecture, although Liam Mulhall, founder and CEO of Brewtopia, worked for years at Red Hat, a successful open source software company.

He, along with a number of friends, came up with the idea of starting a brewery, but one that would always present all decisions to a “community.” What originally began as a group of friends and acquaintances grew into a global community of around thirty thousand beer lovers and connoisseurs. With the internet as its most important tool, a process was initiated in which the community made, one after another, all the necessary decisions. The active members were rewarded for their involvement with certificates that would entitle them to shares if the company ever applied for stock market registration. The community played a prominent role in financing the company, developing the product and marketing the brand. After a period of development, the company actually began to brew and sell beer for the Australian market. In keeping with the notion of open source, a share certificate is given away with every case of beer sold.

Brewtopia beer is an absolutely commonplace, unpretentious beer:

**Just beer to get drunk on**

No crap, no fancy crap about “imported hops,” “first crop barley,” or sweaty blokes hard at it in the coalmines. It’s beer. You get drunk, fall over, start a fight, and mozy on out of town.

Brewtopia

The parade of Brewtopia’s innovations has continued following its market launch. For example, they make it possible for a customer to produce a case of beer according to personal specifications and including a customized label.

In the spring of 2006, the Brewtopia Company did, in fact, become listed on the Australian stock exchange, whereupon the members of the community (and the beer drinkers) became the shareholders. The consumers became the owners, and in fact the producers, just as we have seen in our other examples.

Brewtopia does not bother its customers with expensive advertisements and sales pitches. The most important thing is that the consumer becomes involved with the brand and he (or she) not only tells others how good the beer is but also how nice it is to be a part of the community.



Brewtopia inspires confidence by immediately offering a refund to anyone who is not satisfied. The only condition is that the dissatisfied customer must explain why the beer was found wanting. Of course, feedback of more positive customer experiences is also valued.



Figure 2.17: Grab your chance while you can and become a brewery owner

To further increase brand trust, Brewtopia's pricing strategy is also open. The more beer that is sold, the more cheaply it can be produced. And the cheaper the production costs, the lower the retail price. The Brewtopia site indicates that a case of beer originally cost Aus\$49.99. Then it went to 47.99, still later to 44.98, 42.88 and, in August 2006, the proposed price was as low as Aus\$36.98. The calculations are made based on a defined profit margin: 28 percent.

For skeptics who think that Brewtopia is nothing more than a clever advertising stunt, the company has issued the following statement: "Brewtopia. The only beer company Built by the People for the People." The founders actually needed the investment of consumers to set up the company and they are completely convinced that the open source model of operating a company is the best there is. They are proud that the company is now being used as a case study by Microsoft, *Fast Company*, and the *Financial Review*.

"We accidentally stumbled on what we believe is the best way to run a business – give the customers the reins. It's worked well enough so far that we've even been case studied by Microsoft, *Fast Company* & *Financial Review*!"  
(More on Brewtopia in the Foreword on page 9-11.)

## 2.14 Open innovation at Boeing

The Open Source Space Program of Charles Gerlach, which we discussed earlier, is not yet a reality. In the meantime, Boeing has been incorporating outside knowledge into the building of new aircraft. Boeing made an appeal for help to co-design the new Boeing 787 and supported the request on its website. Anyone could join the World Design Team (WDT) for the new airplane. In total, there are 120,000 registered members. In an interview from 2005, Klaus Brauer the project director states that members of the WDT are mostly concerned with comfort: How is the humidity? Is there wireless internet onboard? Are the washrooms spacious enough? What are the blinds in the windows like? However, we also read that in other discussions with Jeff Hawk, the Director of Certification at Boeing, former Boeing engineers are involved in the discussion and pose critical questions – about the hydraulic system, for example. The WDT initiative is interesting because it uses the experiences of a large group of professional amateurs – or “Pro-Ams” for short. At the same time, a person joining the WDT and visiting the site for the first time might conclude that strong marketing interests underlie it. There is a great deal of advertising for the new plane, there are attractive screen-savers, but there is little opportunity for interaction. For that matter, it is also unclear if most of the work is already done and the community is now merely being thanked for its services. Even the name given the new aircraft, the 787 Dreamliner, is a WDT invention with clear marketing overtones.

## 2.15 Open innovation in building new computers

In addition to the creativity of the software developers in the Microsoft community, Microsoft calls on the resources of Generation C in another manner. Steve Kanenko, Design Director of Microsoft Windows Hardware Innovation, fired the starting pistol for an ambitious project to bring in other external creativity. Anyone with an idea can participate in designing the PC of the future by sending descriptions, photos and videos to Microsoft, and thereby become eligible to win US\$125,000 in prize money. Kanenko encourages participants above all to be frank and to think freely, but also to keep an eye on applicability. Inspired by *The Experience Economy* by Joseph Pine and James Gilmore, Microsoft has included a white paper about computing experience in the starter kit for the competition.<sup>41</sup>

In November 2006, the public chose a winner. The people’s choice received US\$25,000, while the jury’s preference and the personal choice of Bill Gates were each given US\$50,000. In the categories of “Personal Productivity,” “Entertainment,” “Communication & Mobility,” and “Living



and Lifestyle,” the creative generation can indulge itself by providing Microsoft with a helping hand to generate interesting new product ideas.

## Open source hardware

Innovation of an entirely different order is behind the announcement by Jonathan Schwartz, the COO of Sun Microsystems, that the design of the UltraSPARC processor has been turned over to open source. At the January 2006 Open Source Business Conference in San Francisco, Schwartz said that process design would be performed under a GPL license. This is the same open source license under which Linux became available.

“Open source is not just about software. Freedom is not just about software. It’s going to come to hardware, and we’re going to drive that.”

Jonathan Schwartz, COO Sun Microsystems

With this move, Sun wants to encourage the open source community to improve the software and to write better applications for the processor. However, according to IDC, this move also makes it possible for parties in India and China to acquire knowledge to pirate the technology and to clone the processor.<sup>42</sup>



Figure 2.18: Sun open sources its UltraSPARC processor

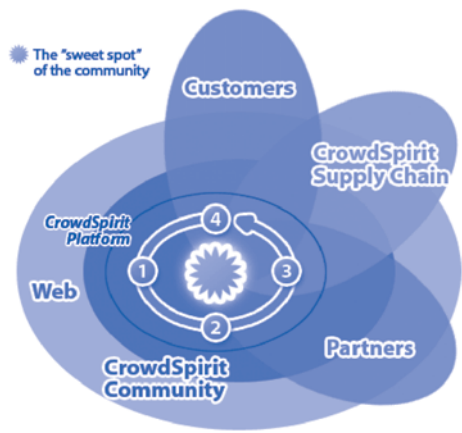
2.16 Open innovation à la CrowdSpirit



Lionel David worked for a company that earned billions from consumer-product sales. His company had discovered that the employee was a valuable source of innovation. Almost every day there was another new “call” for new ideas. But the speed with which ideas were implemented was frustrating to him. The organization was bureaucratic and unwieldy, and Lionel was never given a chance. So he left. Nothing new so far, as entrepreneurial spirit has persisted for centuries in places where it was not nurtured. But what makes this contemporary tale different is that the rules of the game have changed. It has become simpler to challenge billion-dollar companies with new innovative products.

Lionel set up “CrowdSpirit,”<sup>43</sup> which he labels the first company that crowdsources electronic products. The company has a unique formula: Partners for Production. It consists of a network of distributors who collaborate with the internet crowd to make new products.

The Big Picture

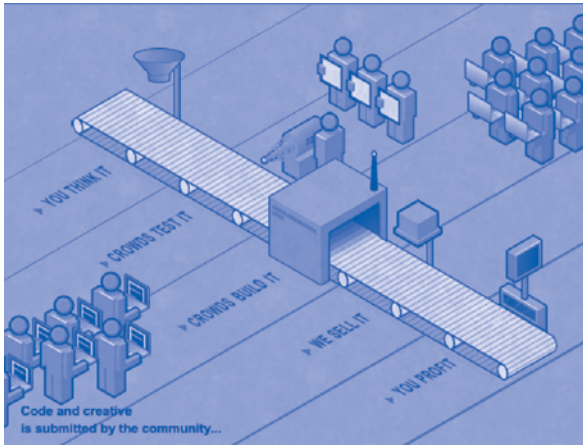


CrowdSpirit: The community is the sweet spot

The nature of any particular innovation does not really matter, so long as it is a consumer product. The product therefore determines the crowd, which is an extreme form of crowdsourcing.

We see something similar at Cambrian House.<sup>44</sup> The people there talk about “crowdsourced software.” The company does not need any distribution channels or any real partners other than the internet crowd itself.

You come up with the product; the crowds test and build it; Cambrian House sells it, and you share in the profits. The steps from invention to cashing-in are presented on the Cambrian House site:



Crowdsourcing from ideas to production at Cambrian House

The “Chumby”<sup>45</sup> is a concrete example of a consumer product that could involve activities similar to those of CrowdSpirit and Cambrian House. What is a Chumby? Practically speaking, it is a clock radio with a wireless connection to the internet. All sorts of things can be built from it or added to it. It is something that was made for hacking and from which something different and more attractive can also be made.



However, the Chumby is also a symbol of all the opportunities that open innovation and crowdsourcing have to offer. The internet makes it possible to exchange ideas and manufacture new products. A half-baked object such as the Chumby can be posted on the internet and people are given the chance to tinker with it. New ideas are generated and introduced in networks, like CrowdSpirit. Many roads lead to Rome. One thing has become perfectly clear: the corporate establishment, the multinationals, are making every effort to participate in open innovation à la CrowdSpirit.

## 2.17 Open innovation at Wikipedia

The open source Wikipedia encyclopedia is one of the largest volunteer networks in the world. There are more than 350,000 registered “Wikipedians: people who have registered with Wikipedia in order to contribute information. In May 2005, the hard core amounted to around 57,000 individuals who had made a contribution ten or more times.<sup>46</sup> One year earlier, this figure was under 20,000. In volume, Wikipedia has long since surpassed the Encyclopedia Britannica. Halfway through 2005, Wikipedia contained four times as many words as its commercial counterpart. It has taken only four years to reach this mark, and Wikipedia is still doubling in size every year.

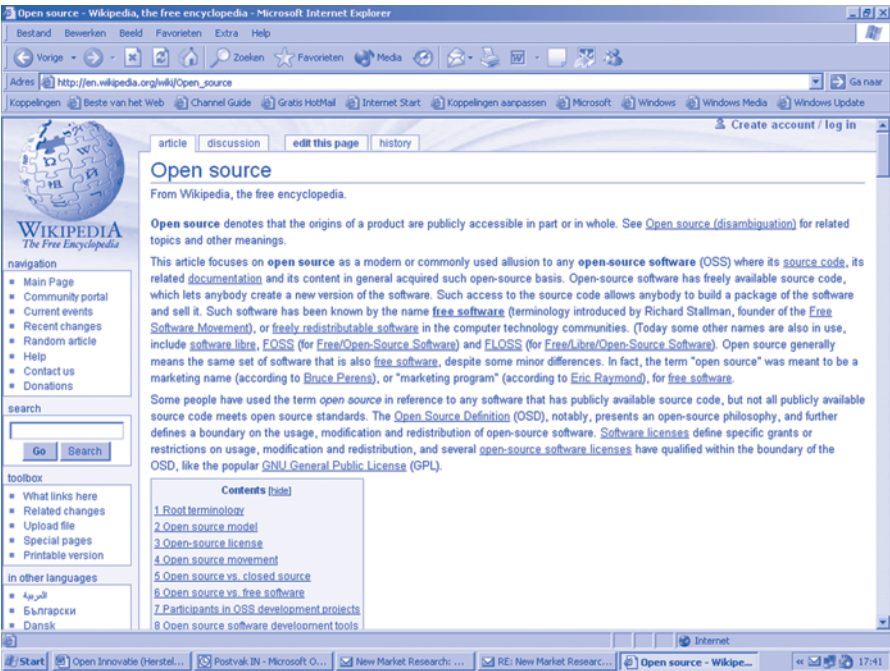


Figure 2.19: Wikipedia is compiled by a community of 350,000 volunteers

Wikipedia only has one employee to keep the servers in operation. Income comes from donations, and the information is stored at the Yahoo machine park. The Wikipedia model is the one that most resembles pure open source software practice as pioneered by Apache and Linux. There is no profit motive. But can an encyclopedia written by volunteers withstand the test of critical appraisal? Is it not overloaded with mediocre contributions and worse? Of course, that is a risk, but the quality control has been quite successful. Research has shown that incorrect information is removed within an average of 17 minutes by one of the 500 administra-

tors. However, these rapid actions cannot guarantee that the information on the rest of the pages is accurate.

Just as with open source software development, peer review is the way to test that the information is reliable. Jokes are made about the quality of Wikipedia by, for example, comparing it to a public toilet: you never know who was the last one sitting there. Such commentary is already included in Wikipedia itself under the heading “Why Wikipedia is not so great,”<sup>47</sup> along with a response to the commentary. The criticism ranges from language use, which is a mishmash of all sorts of English (American, Australian, British, and translated English) to the manner in which improvements are made.

### **Wikipedia is about as good as a classic encyclopedia**

Wikipedia has been proven to be of good quality. In December 2005, research published in *Nature* established that the quality of Wikipedia roughly matched that of the Encyclopedia Britannica. The examination focused on the accuracy of the scientific articles in both encyclopedias. The average Wikipedia article contained four errors. The Encyclopedia Britannica had an average of three. Even more important was the observation that both encyclopedias are plagued by an equal amount of serious errors. In the entire research, four serious errors were found in both the Wikipedia and the Encyclopedia Britannica. Wikipedia suffered slightly more frequently from minor errors.<sup>48</sup>

**Wikinews** A new project was begun in 2004 called *Wikinews*, which reported the news in the same manner as *OhmyNews*. After an official vote that took place between 22 October 2004 and 12 November 2004, [wikinews.org](http://wikinews.org) was launched. The winning *Wikinews* logo was developed by “Neitran,” a German Wikipedia. There is a *Wikinews* manifesto and copyright rules. Erik Moeller, the initiator of *Wikinews*, sees *Wikinews* primarily as a more independent medium. In contrast to *OhmyNews*, it has no advertising revenue.

## **2.18 Conclusions**

This chapter has presented many different examples collectively demonstrating that a new mode of production is emerging, one involving receptivity to external influences and, in particular, the use of expertise and insights from end users/consumers in the innovation process.

We began this chapter writing “innovation” in both capital and small letters. Innovation writ large was identified as “disruptive.” Although open source lessons do not have to be disruptive *per se*, it is certainly interesting

to look at the examples in this way in order to see if there's really anything significant going on. We have compiled a list containing a small cross-section of all the examples with an indication of their effects.

At this point we would have to conclude that only a limited portion of the examples are in the nature of the disruptive innovations described by Christensen. But this isn't so important. It is much more important to recognize traces of open source in numerous sectors and to understand the great promise behind these widespread successes. This development will probably snowball as programming, in both a strict and a wider sense, is becoming accessible to increasing numbers of people. The increasing familiarity with and growing ease of digital involvement will enable new domains to develop more quickly. As both instrumental and primary consequences of this involvement, the volume and impact of open source on business innovation in other economic sectors will grow. The key influences will involve open source culture and open innovation.

Surfkites	A new market player is able to introduce a better and cheaper kite without investing in R&D.
Wikipedia	An encyclopedia compiled by volunteers can compete with the best encyclopedias in the world.
OhmyNews	A newspaper made by volunteers helps the new president get elected to office.
Lead users	3M records an increase in sales of us\$146 million due to a better lead-user focus.
Converse sneakers	Converse has consumers make their own advertising and registers a growth in sales of 12 percent.
Audi	Audi involves customers in the production of commercials and increases the effectiveness of its campaign by 80 percent.
Amazon	10 million visits are made daily to the Amazon site as a result of webservices on other sites.
Threadless	1,500 free T-shirt designs every week.
Lego	Just under 6,000 products have been designed by the Lego community and are being sold by Lego.
Toyota	Production was restored within two days of adopting an approach similar to open source.
Innocentive	A 250-percent yield from participation in an innovation network.
Chicagocrime	New information that was not previously available in this way.
CrowdSpirit	CrowdSpirit will enable you to decide which electronic products you would like to find in your favorite electronic retailer.
Cambrian House	The self-proclaimed "Home of Crowdsourcing".
Chumby	Chumby displays useful and entertaining information from the web: news, photos, music, celebrity gossip, weather, box scores, blogs – using your wireless Internet connection.

## How do these innovations occur?

A second observation about the disruptive character of these examples concerns the degree to which these disruptions occur in the manner that Christensen predicts. Specifically, Christensen suggests that the original rudimentary functionality of an open source production process provides fertile ground for what, in effect, then become disruptive innovations. In the case of such products as Wikipedia or *OhmyNews*, we can certainly see that there is something to this view. Initially, existing market players did not lose any sleep over these marginal products. Only when their quality began to improve did it become clear that this new mode of production could be significant.

However, Converse, Audi, Threadless and Amazon were all fully involved in open innovation in a commercial way right from the start. The surfkite would not have taken over the market had its performance not been superior to the market leader's; in that example, the product functionality was demonstrably better from the beginning. And mash-ups such as Chicagocrime.com must also satisfy concerns about meager product quality precisely through the creativity of making something new from the combination of existing things.

## Inspiration from open source

This brings us to one final reflection before we distinguish a number of tendencies illustrated by the examples. What is the ultimate cause of the success? Is it the special licenses? The users who are driving it? Internet and the virtual worksites? Self-determination and intrinsic motivation? The intersection between open and closed? Or clever management due to transparency and a meritocratic organization? We began this book with these elementary characteristics of open source. A definitive answer to the question cannot be given. It is also not so surprising that Brewtopia, the open source beer brand, is so often cited as a case study. They have pulled out all the stops and decided, "We will do everything in a manner that is as open(-source) as possible and we will see what happens." Now, this wouldn't seem to be the most reasonable advice for other companies. However, it makes the clear point that although open source has potential, we still do not know how it is going to develop.

There is one thing that we are able to say with certainty: the ease with which people can participate in innovation will only increase. As we saw in the case of the open source shoes from John Fluevog, the underside of a beer mat is enough. Participation can be in ordinary language, as we saw in the case of Wikipedia. And the other, more complicated languages all have the feature that they are becoming much simpler: seven-year-olds design new Lego toys using CAD software.



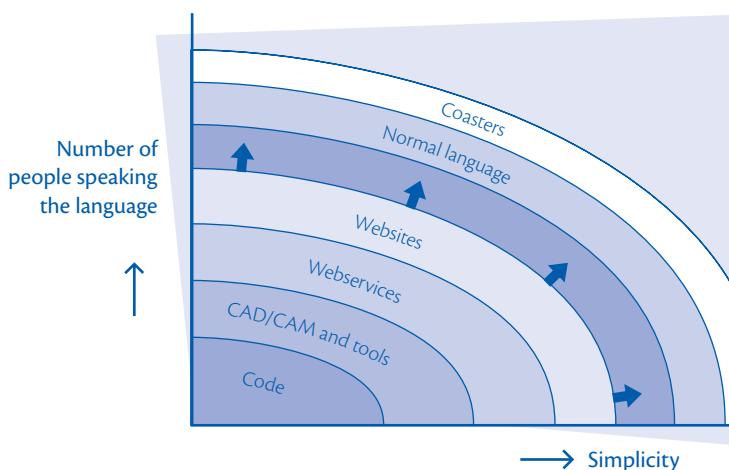


Figure 2.20: Everyone can make a sketch, send in an idea or participate in some other manner; with open innovation, this involvement is becoming increasingly easier

The languages with which open innovation can be realized are shown on the horizontal axis in Figure 2.20. The simplicity of the language of communication increases as we move right. The simpler the language, the greater the number of people who can speak it, which is indicated on the vertical axis. The arrows in the gray area indicate that it is becoming increasingly easy to be creative by means of software and the internet. Open innovation gets off the ground because more and more people are digitally competent; the internet, webservices and software enable them to collaborate among themselves or with companies.

Publicly accessible source codes, open source, can only be fully realized in a software environment. In “physical” (at least, non-digital) environments, we often only deal with elaborate specifications or perhaps only with rudimentary ideas. The more that companies pursue innovation, and this will happen as more successes become evident, the more motivation there will be for companies to release source codes and tools in order to make them available to an innovative community. The engagement of expert users in innovation, as occurs in the software environment, will increase as a result, because the opportunities to participate are also going to increase. Furthermore, the statement of Eric von Hippel will prove true: innovation is democratizing. Of course, there is a question of how all this will relate to the commercial interests of companies and institutions. With the exception of a small number of non-commercial open-innovation initiatives, most examples have a commercial slant. After reading the three following chapters, it will be clear that the commercial interest of companies is an important pillar supporting the rapid development of open source. What software developers think about the commercial exploitation of their efforts will be explored in Chapter 4.



To conclude, we will end this chapter with six general tendencies that can be derived from the examples discussed above.

### **1 Community products in natural languages**

The first tendency involves products that are made or written in ordinary language. Since no complicated programming knowledge needs to be employed, it is possible for the general public to participate. To a great extent, such products rely on this community involvement. In other words, if the community were no longer to exist, little of the end product would survive (this is less the case in other open business practices). Wikipedia and *OhmyNews* are good examples. Just as in the case of open source software, such products are launched in imitation of something that already exists: an encyclopedia and a newspaper. Although initial concerns about “crummy” quality/functionality may have been valid, it is important to note that *OhmyNews* now belongs to the top five newspapers in Korea, and Wikipedia is no longer inferior to professional encyclopedias. The community is fully self-supporting: one computer with an internet connection is sufficient. The ultimate goal is to collectively produce a good product.

### **2 Open innovation in advertising and marketing strengthens the bond with the customer**

Although the focus of open innovation falls on internalizing the knowledge that exists outside the organization, something else is involved in advertising and marketing. Besides good ideas, open innovation in these domains is primarily concerned with strengthening the bond with the target group. Obviously, any company can involve customers in the production of advertising: examples include Nike, Mercedes and Audi. These companies encourage customer involvement, and not just for reasons concerning any alleged shortage of creativity in advertising agencies or the high rates charged by such agencies. Identification with the brand and the resulting bond with the customer are both enhanced by such activity. The dividing line between marketing and open innovation is extremely thin, as shown in a number of examples. Open innovation concerns marketing in the cases of Threadless, Brewtopia, Jones Soda, and Yahoo. The general manager of Yahoo literally admits that their innovation approach is essential for their R&D and marketing. Lego identifies customer recommendations to others as one of the four important gains from open innovation. There is consequently a spin-off for advertising and marketing resulting from the involvement of outsiders in a company's organization and as an upshot of allowing them to participate in the innovation process. We have remarked on this before, in relation to activating the user-driven innovation that Von Hippel preaches in his recent book *Democratizing Innovation*.

### **3 From digital design to physical product**

If the final product is fully digital, as in the case of software or Wikipedia, a community can set to work on digitally modifying many independent aspects. No one needs to have sufficient resources to pay for machines or material to make the physical product. Open source blueprints for innovation involving physical end products therefore function in a significantly different manner. But the more that things are automated and the more that digital work can be separated from physical activity, the more this digital/physical distinction is blurred. The tendency in this regard has been clearly demarcated, and the tools that support the digital work are becoming progressively better and less expensive. We saw this in the surfkite example. Practical and inexpensive software made it possible to begin working in an open source manner. A process of trial and error could not, however, be avoided in the production of the kite. Material was certainly required for construction and testing, but the investments remained manageable. Budgets can be further reduced as the predictive quality of the digital design increases. Simulation and testing software (and the integration of the two) makes this possible, and again the quality of these is being progressively improved. This leads directly to more open innovation and to more open source innovation. In other fields, such as bio-computing, a great deal of progress has already been made. Lego, which has built up a strong lead in this regard, states openly, "We are not an exceptional company; any organization can do something similar." What is characteristic and essential in these examples is the fact that the software used in programming does not have to be open source. The focus is on the end result, product or medicine that is produced in a community to be entrusted to the public domain.

### **4 Webservices blur the boundaries between communities and organizations**

Google, Amazon and Yahoo provide developers with the opportunity to easily devise and offer new services with the help of webservices. The developers may be called "associates" in some cases, but they are in fact one and the same thing. Companies like Google open themselves up to anyone who wishes to do something with one of their services, because they know that their own ingenuity is limited. Consequently, they receive webservices and ideas from the community. At the same time, the energetic web-service strategy means that these organizations have to know how such a community works and how they can tap into it. When ten million visitors call at Amazon every day as a result of accessing their websites through other sites, it becomes clear that the boundaries distinguishing the organizations involved are not drawn very sharply. Such interlinking or "interwebbing" is noticeably on the increase. Google, Amazon and Yahoo are the leaders, but the development of webservices, the associated interwebbing and the emergence of service-oriented architectures appear to be just beginning.

## 5 Laboratories, people and companies link up

Taking the initiative from large companies, innovation networks have been created in which individuals and other companies participate. Such organizations as Innocentive, YourEncore and NineSigma are mining outside knowledge in response to the realization that not all smart people work for a single company. It is interesting to see that small laboratories in people's homes can have an effect on the innovations adopted by large companies. Using software simulation for experiments may well be the future; but for the time being, wet labs are still able to compete. There is also a certain degree of self-determination evident in the open source community. The "seekers" of specific types of knowledge make themselves known, and the "solvers" choose the problem that they would like to solve. However, the intellectual property of the findings is, in general, quickly fenced in. We saw that in the case of Boeing. Its own community, the World Design Team, was primarily set up to find clever ideas for the company itself, even if parts of the discussions were made public and accessible to everyone.

## 6 Internalization of open source culture

As an alternative to profitably utilizing outside knowledge and expertise, there are also open source blueprints that can be successfully applied inside an organization. In a time of crisis, Toyota resorted to a mode of production resembling open source. This company's assembly lines incorporate strong elements of the self-determination that is characteristic of open source production. Ultimately, Toyota adopts such practices in order to deal more effectively with problems and, at the same time, to reduce expenditures on consultation and the formulation of top-down plans. Similarly, it is interesting to note that something is also happening inside Lego as a result of customers becoming involved in the design process. Collaboration has become a more regular feature of the business culture at Lego. In Chapter 5, we will see that software producers are also seeking salvation in an open source culture, as they expect far more from such practices than they might achieve with a purely hierarchical mode of production.

The final conclusion to this chapter cannot, of course, be anything other than the observation that open innovation stands or falls with the community that congregates around a product, service, process, organization, brand or market.

### Notes to Chapter 2

1. Schumpeter, J., *The Theory of Economic Development*, Harvard University Press, Cambridge Massachusetts, 1934.
2. Christensen, C., *Capturing the Upside*, OSBC, 2004, [www.itconversations.com/transcripts/135/transcript-print135-1.html](http://www.itconversations.com/transcripts/135/transcript-print135-1.html).

3. O'Reilly, T., *Ten Myths about Open Source*, 1999, [opensource.oreilly.com/newms/myths\\_1199.html](http://opensource.oreilly.com/newms/myths_1199.html).
4. Weber, S., *The Success of Open Source*, Harvard University Press, 2004.
5. [www.eweek.com/article/os1895,1896588,00.asp](http://www.eweek.com/article/os1895,1896588,00.asp).
6. Prahalad, C.K. & V. Ramaswamy, *The Future of Competition*, Harvard Business School Press, 2004.
7. Prahalad, C.K. & V. Ramaswamy, "How to Put Your Customers To Work," *CIO*, June 8, 2004, [www.cio.com.au/index.php/id;1736056082;fp;512;fpid;184745984](http://www.cio.com.au/index.php/id;1736056082;fp;512;fpid;184745984).
8. Mulgan, G., T. Steinberg & O. Salem, *Wide Open; Open Source Methods and Their Future Potential*, Demos, 2005.
9. IBM, "The Next Big Thing No Longer Exists," *CNET*, 15 March 2006, [news.com.com/ibm+The+next+big+thing+no+longer+exists/2100-1008\\_3-6050056.html](http://news.com.com/ibm+The+next+big+thing+no+longer+exists/2100-1008_3-6050056.html).
10. Hippel, E. Von, *Democratizing Innovation*, MIT Press, 2005.
11. [www.iscb.org/overton.shtml](http://www.iscb.org/overton.shtml).
12. See [www.factory-simulation.com](http://www.factory-simulation.com).
13. Magnusson, P.S., "The Virtual Test Lab," *IEEE Computer*, May 2005, Vol. 38, 5; p. 95-97.
14. [www.adtmag.com/print.asp?id=12020](http://www.adtmag.com/print.asp?id=12020).
15. Gerlach, C.D., *Building an Open Source Space Program*, 2005, [isdc2005.xisp.net/~kmiller/isdc\\_archive/fileDownload.php/?link=fileSelect&file\\_id=139](http://isdc2005.xisp.net/~kmiller/isdc_archive/fileDownload.php/?link=fileSelect&file_id=139).
16. [www.fluevog.com/files\\_2/os-1.html](http://www.fluevog.com/files_2/os-1.html).
17. Florida, R., *The Rise of the Creative Class, and How It's Transforming Work, Leisure, Community and Everyday Life*, Basis Books, 2002.
18. Lenhart, A., J. Horrigan & D. Fallows, "Content Creation Online," *Pew Internet & American Life*, February 29, 2004, [www.pewinternet.org/pdfs/pip\\_Content\\_Creation\\_Report.pdf](http://www.pewinternet.org/pdfs/pip_Content_Creation_Report.pdf).
19. Nuttall, C., "Way of the Web: Start-Ups Map the Route as Big Rivals Get Microsoft in Their Sights," *Financial Times*, November 17, 2005, p. 15.
20. Gens, F., *IDC Predictions 2006: It's Gut-Check Time, As Disruptive Business Models Gain Traction*, December 2005, [www.linuxelectrons.com/article.php/20051201222935677](http://www.linuxelectrons.com/article.php/20051201222935677).
21. [mcmanus.typepad.com/grind/2004/07/write\\_opensourc.html](http://mcmanus.typepad.com/grind/2004/07/write_opensourc.html).
22. Singel R., "Map Hacks on Crack," *Wired*, July 2, 2005, [www.wired.com/news/technology/0,1282,68071,00.html](http://www.wired.com/news/technology/0,1282,68071,00.html).
23. "Mix, Match, and Mutate," *BusinessWeek*, July 25, 2005, [www.businessweek.com/magazine/content/05\\_30/b3944108\\_mzo63.htm](http://www.businessweek.com/magazine/content/05_30/b3944108_mzo63.htm).
24. [www.nivi.com/blog/article/greasemonkey-and-business-models](http://www.nivi.com/blog/article/greasemonkey-and-business-models).
25. "Van webwinkel tot softwareplatform," *Computable*, April 22, 2005.
26. See Nasa clickworkers at [clickworkers.arc.nasa.gov/crater-marking](http://clickworkers.arc.nasa.gov/crater-marking).
27. "The Worlds Most Innovative Companies," *BusinessWeek*, April 24, 2006, [www.businessweek.com/magazine/content/06\\_17/b3981401.htm](http://www.businessweek.com/magazine/content/06_17/b3981401.htm). Also

- see “Understanding Toyota’s Innovation Factory”, [www.businessinnovationinsider.com/2006/10/understanding\\_toyotas\\_innovati.php](http://www.businessinnovationinsider.com/2006/10/understanding_toyotas_innovati.php).
28. Evans, P. & B. Wolf, “Collaboration Rules,” *Harvard Business Review*, July/August 2005.
  29. Chesbrough, H.W., *Open Innovation: The New Imperative for Creating and Profiting from Technology*, Harvard Business School Press, 2003.
  30. Klee, K., “Grand Opening: Procter & Gamble Once Kept R&D Close to the Vest. Now Outlicensing is Driving Growth,” *IP Law & Business*, February 2005.
  31. *The Wall Street Transcript*, 2005, [www.innocentive.com/about/media/200504\\_wsj\\_Interview.pdf](http://www.innocentive.com/about/media/200504_wsj_Interview.pdf).
  32. [english.ohmynews.com](http://english.ohmynews.com).
  33. CNN, *Korean Bloggers Making a Difference*, March 31, 2005, [www.cnn.com/2005/tech/03/31/spark.ohmynews](http://www.cnn.com/2005/tech/03/31/spark.ohmynews).
  34. Schroeder, C., “Is This the Future of Journalism?,” *Newsweek*, June 18, 2004.
  35. Juha-Pekka Raeste, “In South Korea, Every Citizen Is a Reporter,” *Helsingin Sanomat*, January 8, 2005.
  36. Cherkoff, J. *What is Open Source Marketing?*, [www.changethis.com/14.OpenSourceMktg](http://www.changethis.com/14.OpenSourceMktg).
  37. Hof, R.D., “Advertising Of, By, and For the People,” *BusinessWeek*, July 25, 2005.
  38. Kiley, D., “Audi’s Art of The Heist Captured Leads,” *iMediaconnection*, July 26, 2005, [imediainconnection.com/content/6386.asp](http://imediainconnection.com/content/6386.asp).
  39. [www.conversegallery.com](http://www.conversegallery.com).
  40. The commercial can be viewed at [www.centraalbeheer.nl/cbi/cb/corporate/fun/commercials.jsp](http://www.centraalbeheer.nl/cbi/cb/corporate/fun/commercials.jsp).
  41. Allchin, J. & N. Kano, *Experience Computing*, Microsoft, April 22, 2005. [www.startsomethingpc.com/EntryKitFiles/Experience%20Computing%20Overview%20Whitepaper.pdf](http://www.startsomethingpc.com/EntryKitFiles/Experience%20Computing%20Overview%20Whitepaper.pdf).
  42. Shankland, S., “Sun Releases SPARC Specs in Linux Love Bid,” *CNet*, February 15, 2006, [news.zdnet.co.uk/hardware/chips/0,39020354,39252513,00.htm](http://news.zdnet.co.uk/hardware/chips/0,39020354,39252513,00.htm).
  43. [www.crowdspirit.org](http://www.crowdspirit.org)
  44. [www.cambrianhouse.com](http://www.cambrianhouse.com)
  45. [www.chumby.com](http://www.chumby.com)
  46. [en.wikipedia.org/wikistats/en/ChartsWikipediazz.htm](http://en.wikipedia.org/wikistats/en/ChartsWikipediazz.htm).
  47. [en.wikipedia.org/wiki/Why\\_Wikipedia\\_is\\_not\\_so\\_great](http://en.wikipedia.org/wiki/Why_Wikipedia_is_not_so_great).
  48. Giles, J., “Internet encyclopaedias go head to head,” *Nature* 438, 900-901, December 2005, [www.nature.com/nature/journal/v438/n7070/full/438900a.html](http://www.nature.com/nature/journal/v438/n7070/full/438900a.html).



### 3 The software market has taken the lead

This chapter will examine the effects of open source on the software market. Not surprisingly, it is a market in which many players engage, to a certain extent, in open innovation. In this case, open innovation is open source innovation. Even a company reputed to be completely closed, such as Microsoft, has taken many more steps toward open innovation than have many companies outside the software sector. We will describe a number of organizations that use open source software to generate new business, along with a selection of traditional software companies. The two together have rapidly transformed the market with their open-source-based business models. Measured in terms of the impact of open source, the software market is demonstrably ahead of the pack in this regard. In no other sector is market innovation performed so extensively. The discussion of open-source-inspired innovation in this chapter will be concluded with a series of reflections on open innovation outside the software sector.

#### **Progressively more open source business**

Nowadays, not a single company in the software industry can ignore open source. This is certainly related to the growth in the number of applications that have come into the public domain. They can all be used for business purposes by anyone wishing to do so. On the demand side, there are increasing numbers of organizations that want to acquire open source software. Producers are all adopting open source; consequently, all software companies have “open” strategies. This is one difference from what we described in the first chapter. Most companies outside the software industry do not have such a strategy, or at least not yet.

We will begin this chapter by listing the seven distinctive types of open source strategies implemented by software producers, ranging from optimization to embedding. Often they occur in combination.

Open source strategy	Description of the supplier
Optimization	Lower costs while maintaining reliability and adding robust open source software: for example, SAP or Linux.
Dual licensing	Open source in a free version and a professional edition: for example, SugarCRM and MySQL.
Consultation	Advice about which open source software is the most suitable: for example, based on the Open Source Maturity Model from Navica.
Patronage	Open source community aid: for example, a great many of the Linux kernel developers are employed by IBM.
Subscription	Companies deliver a package of open source products and associated services: for example, Red Hat and Covalent.
Hosting	Internet companies operate on open source software: for example, Amazon and Google.
Embedding	Open source software is used in devices: for example, Linux smartphones (see <a href="http://www.linuxdevices.com">www.linuxdevices.com</a> ).

## Tricks for making money

There is a great drive to earn money from open source software. It is particularly keen because anyone can compete using products belonging to the public domain. First of all, there is money to be made from support services: much open source software is not known for its ease of installation. Various companies take advantage of this factor by packaging the original product more attractively and introducing it on the market at a commercial price. Under the terms of a service contract, they also ensure that new updates are implemented promptly.

Income can also be earned by eliminating risks. For instance, hardware suppliers guarantee the operation of their systems provided that the Linux operating system is acquired from a given open source supplier. This is one of the reasons why Red Hat is much in demand as a Linux supplier. Overall management of open source risk is offered by Open Source Risk Management ([www.osriskmanagement.com](http://www.osriskmanagement.com)).

In the “expenses” column, there are savings. Up to now, open source companies have not required large sales and marketing budgets to be successful, and they save substantial costs in research and development due to the contributions of the community.

Finally, it is important to remember that the willingness of traditional IT companies to support open source software is frequently related to their desire to curtail the power of other players. Earning money and taking the wind out of the sails of the competition naturally go well together.



It is for this strategic reason that companies such as IBM and HP provide patronage.

### **Open source: a snapshot of the IT world**

Almost every day there is something new to note about open source. Consequently, this chapter can only provide a portrait at a point in time. What was closed yesterday might be suddenly turned into open source tomorrow. And a closed source operator with a passive open source policy could suddenly make a takeover move tomorrow and have to absorb an open source company. The market is continuously in motion, and development occurs rapidly. Hardware suppliers have, so to speak, the least to lose or the most to win. Open source software primarily offers them alternatives for strengthening their relationship with customers. Microsoft has the least affinity with the business of open source because it supplies nearly the entire software stack. The company is particularly apprehensive about competition from Linux affecting its important cash cow, Windows. Other players often rally their forces in order to mount a collective assault on the market. Of course, there are also alliances between established producers and newcomers, given the segmentation of software manufacturers, packagers, resellers, IT service providers, or combinations of them. The alliances may go as far that the one (traditional software company) buys out the other (the open source company). Below, we will describe developments in the sector by referring to a number of producers. We do not intend to give a complete picture of the software market, merely use these examples to illustrate how the open source model affects strategy.

## **3.1 Open source has transformed the software portfolio**

Open source software has now completely shed its anomalous, eccentric and “anarchic” status. In 1998 this term was first used to tag it as aberrant, although somewhere around 2004 a change seems to have occurred. As evidence of this transition, IDC published a report at the end of 2004 with the revealing title “Worldwide Linux 2004-2008 Forecast: Moving from Niche to Mainstream.” In mid 2005, Microsoft even announced that its Virtual Server product would support Linux, and Sun Microsystems open sourced its Solaris operating system. Nowadays, there is an open source alternative for nearly every business need. In most cases the alternatives are less “scaleable” (but are nevertheless capable of it), and they range from databases (MySQL) to complete applications, such as the Compiere ERP package and SugarCRM. Expectations are that open source software products will continue to bring about further changes in the

software world. This is illustrated by Figure 3.1, which specifies the percentage of open source software employed in seven IT categories.

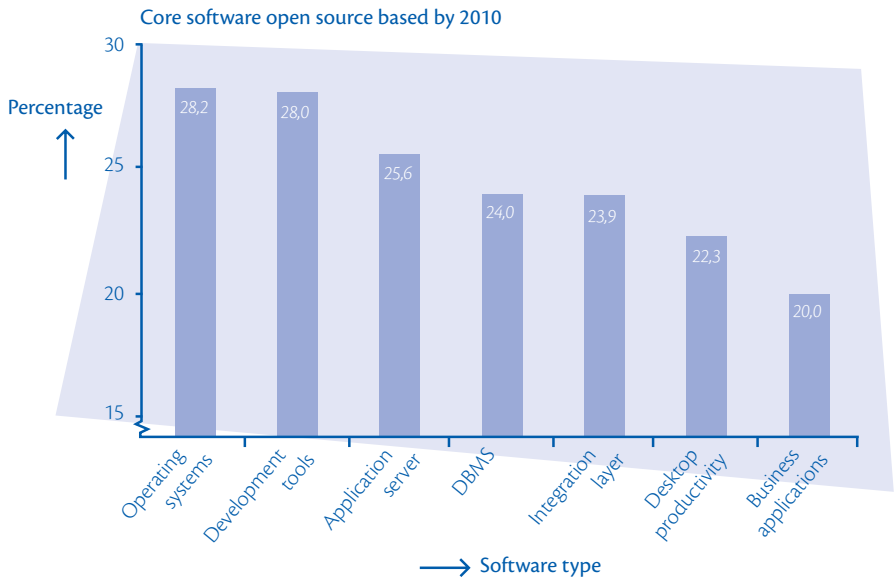


Figure 3.1: Open source pervades all segments (Saugatruk, December 2005<sup>1</sup>)

At the end of November 2005, the American business journal *CIO Insight* reported that open source software had become a strategic choice for organizations (at least in the US). The following brief survey shows the extent of this shift:

**Who embraces open source and why?**

- 81% of companies have access to open source software or are considering the possibility of using it.
- 72% are planning to expand the use of it.
- 64% say that open source has provided them with a competitive advantage.
- 65% say that open source has stimulated the innovation in the company.
- 67% choose open source primarily because of costs.

CIOInsight.com, November 2005<sup>2</sup>

Originally, and we are speaking about less than ten years ago, the open-source world was almost completely divorced from the rest of the IT industry. It was cut off from IT service providers (Capgemini, Atos Origin, Accenture, etc.), from hardware manufacturers (IBM, Sun, HP, Dell, etc.) and from the software establishment (IBM, Microsoft, Oracle, SAP, etc.) Ten years ago, there was hardly any notion of open source software producers like those of today (Red Hat, Suse, Novell, Compiere, SugarCRM

and MySQL). The open source software community (avant la lettre!) manufactured “free software” and bore, depending on the observer, an anarchic, libertine, democratic, socialist or communist ideological stigma. This “free software” had absolutely no credibility as an alternative to commercial, proprietary-driven operating software, databases, ERP systems, *etc.* Computers operating on Linux were only for “hackers,” at that time a nickname that had nothing to do with the current more pejorative term meaning “computer criminals.”

The open source community was originally a somewhat isolated club of activists who felt that computer software should not be a source of sky-high profits but should actually be public property. This would be fairer, because the world was becoming more dependent on software systems, and unwanted and unnecessary features should be appropriately eliminated in an “egalitarian” open source world. “Innovation” in the software field would go wrong less quickly. After all, it was the age of new versions (in particular of UNIX and PC software packages), which were being generated in rapid succession. Everyone, including the software manufacturers themselves, thought that this was ultimately a rather unhealthy situation, obviously focused on ever-increasing profit. Consequently, the free-software movement certainly had a point, insofar as this oversupply of functionality was concerned.

When the free-software movement mutated into open source software in 1998, serious competition, specifically aimed at proprietary software companies, had become an established fact. Today, we mention free software and open source software in the same breath and see the acronyms F/OSS or FL/OSS everywhere – the “L” standing for “Libre” in order to indicate that we must understand “free” in the sense of “freedom” rather than in the sense of “free of charge.” The serious competition dates from the same year, all beginning with Linux and Netscape, which was open sourced under the name “Mozilla.”

At present, not yet ten years later, the situation has been reversed. What was once describable as “almost completely divorced from the rest of the IT industry” has been replaced by the “fully integrated” current state of marriage. In addition, open source software has become strongly commercialized, which is viewed as a despicable development by some but is regarded by others as a move that guarantees continuity through the healthy interplay of supply and demand. Open source software has completely changed the IT playing field. For instance, a former network software company such as Novell now focuses on open source software. The same holds true for IBM, Sun, HP and others. Software openness and true open source software products have attained a definitive and noteworthy place in the contemporary IT landscape. Undoubtedly, the price advantage of open source products in recent years has helped to create a more positive image of the IT sector among customers.

## 3.2 The business model of the original open source organizations

Many open sourcers sell “security”: the security that you have the right version of the product, that you will receive updates and patches on time, that you will not be ensnared in legal disputes, that the software is easy to install, and especially the security that you can contact someone whenever there is a problem. This is especially notable in the case of Linux products. Anyone can download and install them, but it is not an easy undertaking. Red Hat and Novell (Suse, Ximian) are two companies that capitalize on these concerns. Since Linux is, in fact, threatened by liability to third parties, indemnification is also a valued service. For this reason, new companies have come into existence to act as a sort of interface between a particular open source community and its users.

New companies such as Yggdrasil Computing, Ximian, Debian, Suse and Red Hat are capable of providing advice about the more service-oriented open source issues. They focus on the “market introduction” of open source. Their approach is above all pragmatic; they are not driven by any idealistic notions such as those propagated by Richard Stallman’s Free Software Foundation. These companies just want to make money by providing good service.

Anyone questioning the success of an open source company will often hear, in response, the number of times that the program has been downloaded (for example, 10,000 times a month). Does such activity generate revenue, though. The supplier hopes that a portion of these downloads will result in customers willing to pay for services. Larry Augustin, an open source entrepreneur right from the start, founder of VA Linux Systems (now VA Software) and member of the Board of Directors of the Open Source Development Lab (OSDL), once calculated what would happen if Siebel, a CRM manufacturer, were to make its software open source. If they had opted for the try-and-buy model of open source programs (first download and see if it works, then perhaps buy it), Siebel would have, first of all, reduced expenditures on marketing and PR. These could have been cut to one quarter of the original amount. After all, the development of expensive advertising campaigns would not have been required and a large number of the well-paid sales personnel would also have been unnecessary. By making greater use of a software development community, Siebel could have saved on R&D as well. And overhead costs would have also strongly decreased in adopting the new model. Augustin’s calculations show that the profitability of open source operations has a lot to do with reduced expenditures. The takeover of Siebel by Oracle means that Augustin’s proposal for Siebel will remain a fiction, but it is certain that proprietary software manufacturers will include these sorts of calculations in considering the possibility of making their software open source.

Although the original open sourcers did not have to invest in costly advertising campaigns and the like, they still had to obtain income from somewhere, even if their costs were lower. The try-and-buy method was primarily applicable to producers offering a dual license: a normal open source license for a limited version of the software and a paid license for the extended version. However, there were other methods for generating supplementary sources of income. Let us now consider how a number of open source organizations managed to earn their wages.

## Apache

The most successful open source product, the Apache webserver (which has a 70 percent market share), is produced by an organization without any commercial drive. Apache is a non-profit organization made up of volunteers. Like other volunteer organizations, Apache operates for the most part on the basis of donations. The United Layer company hosts the site free of charge. HP, Sun and IBM make hardware available. The most important donations consist, however, of the time that volunteers spend working in the Apache community. These volunteers are primarily people that work with Apache while they are on the payroll of other companies. The president of Apache, Brian Behlendorf, earns his income in other capacities: in particular as president of CollabNet, as well as from his lectures and books. Apache does not earn any money from customer support, simply because the organization has chosen not to offer it. Questions from users are initially handled by means of an internet news group. This has worked well, but as is the case with all other important open source programs, there are no commercial Apache support companies.

## MySQL

Database supplier MySQL earns its money solely by selling licenses. It is a so-called dual-license supplier, which means the software is made available under both an open source and a commercial license. The commercial license offers the advantage of more security, support and a service guarantee. MySQL is downloaded 20,000 times a month. One in a thousand downloads results in the purchase of a paid license. Besides licenses, MySQL has two other sources of income: membership in the online services, and franchises for MySQL products and services under the MySQL brand name. Some people in the open source world disapprove of this form of licensing policy, but it is reasonably successful in practice. Annual sales amount to approximately US\$40 million, and double every year. Marten Mickos, the CEO of MySQL, states in an interview with *Business-Week*<sup>3</sup> that future stock market registration is certainly in the realm of

possibility. Although an early takeover by Oracle was rebuffed, Mickos also considers it possible that the organization may eventually become a part of a larger company, on condition that the MySQL name survives.

## **Zend**

This supplier of the PHP web development language earns money from support services and membership contributions to the Zend platform. The Zend services include training, application design and PHP code review. The investors behind Zend are a number of venture capital funders, but also companies such as SAP. The Zend customer database numbers about 4,000 businesses, including Lufthansa, Disney and Wall-street:online. Zend's strategic partners are IBM, SAP, MySQL, Adobe and Sun.

## **Red Hat**

Red Hat delivers Linux software that is certified by IBM, HP and Oracle. This status makes Red Hat profitable. Companies wanting to run Linux on their servers can do so without Red Hat; they can download and install the operating system for free. However, the certification by the suppliers provides some security concerning the proper operation of the software and – even more importantly – indemnity from any potential third-party legal claim based on alleged property rights to the software. This “indemnification” is a very important point. The price of Red Hat Linux also includes a membership that entitles the purchaser to a number of support services, such as regular updates and rectification of errors. Depending on the need, registration in additional service programs is also offered.

## **Novell**

Novell is a veteran player that, in recent years, has undergone a complete transformation taking it in the direction of open source. The company earned just under US\$1 billion from maintenance and service and a quarter billion from licenses. The license income, which mostly results from the old Netware, is declining because the product is nearing the end of its lifecycle. By taking over Suse and Ximian, Novell has acquired a strong position in the market for the Linux desktop in particular (although Suse provides Linux for servers and desktops). In exchange for a license (including upgrade protection and a contract for technical support), Novell indemnifies its customers for any potential third-party

claims based on intellectual property rights. For that matter, the costs of such licenses can amount to as much as 24,000 euros per year per server. Suse earns money from training and consultancy in the field of open source. With the takeover of Ximian, Novell now also delivers desktop applications for Linux, such as text processing and slide presentations – incidentally, these are applications that derive from the Gnome open source project.

## Digium

Digium is a commercial hardware company that does not earn any money from open source software but from selling products that complement use of the software. With Asterisk, its open source product, Digium is seen as a challenger to the established telecommunications industry. The Asterisk software of “voice over IP” telephony is rapidly becoming incredibly popular. This software, which runs on Linux, is being offered by 130 suppliers worldwide. It has the capacity to connect to any make of telephone exchange and enable delivery of VoIP services, including conference calls and voicemail.

The software was originally written by Mark Spencer, one of Digium’s employees, but the code was then supplemented and extensively tested by an open source community. Given that Digium was responsible for designing Asterisk, it can claim that its hardware is more suitable for running it than any other: “Since Digium is the creator of Asterisk, Digium hardware is designed specifically with Asterisk in mind.”

“Spencer says that companies are confronted by the choice of standing on the sidelines or becoming themselves ‘disruptive innovators’. By making Asterisk available as open source under a GPL license, Digium has evidently chosen the second option. The telecom market offers an ideal piece of fertile ground in which to do this, given its extensive customer group and the substantial price difference between proprietary solutions and open source alternatives. The relatively technical character of the telecom market is also helpful. Ultimately, the message from Digium is that its products are tailored to the perspective of the customer, because vendor lock-in is avoided. Or in Spencer’s own words, ‘Open source and the GPL are about the customer first, not the vendor. So you have to adapt to those things.’”

Mark Spencer, 2005<sup>4</sup>



## SugarCRM

SugarCRM is a commercial open source company. It was founded in 2003 by John Roberts, Clint Oran and Jacob Taylor, all three of whom had worked for Epiphany, a CRM supplier with US\$100 million in sales. The company produces CRM software, a product that has special significance because many open source skeptics doubted whether this sort of value-added software could be provided in an open source manner. SugarCRM employs a dual-licensing strategy, just like MySQL. This means that the company earns money from a commercial version of the product: the Pro-version. Pro users pay for extra features and services. For \$40 per user per month, a hosted version is offered, and US\$239 buys a regular license for one user for one year. Help for installation is offered in three-hour blocks. The open source license is a Mozilla variant (Sugar/Mozilla) and therefore not a GPL. To emphasize the open source nature of the company, the company claims to spend only ten percent of the time devoted to new releases on the commercial versions.

Two venture-capital financiers have invested in SugarCRM: Draper Fisher Jurvetson and Walden International. Their investment was based on the number of downloads, which has now risen to 20,000 per month.

Many of the open source players are supported by the established order; IBM, HP, Sun, Adobe, SAP, and Oracle have all provided a helping hand. Assistance has taken various forms: making direct donations of time and money (as in the case of Apache), assuming the legal risks involving copyright claims (as in the case of Red Hat Linux), or forming strategic partnerships (as in the case of Zend). This is, however, insufficient to keep a company's head above water.

Open source organizations have various business models. For example, the money-generating capacity of Apache lies entirely outside of the company itself. Money is earned by exploiting knowledge about open source in another manner. Furthermore, there are players who employ a dual-licensing strategy, offering the market both a free and a paid variant.

SugarCRM is an organization that, in addition to dual licenses, offers a hosted version of the product as well. This offers organizations the opportunity to make use of the open source software without having to actually own it. SugarCRM is, in this way, a direct competitor with on-demand Siebel or CRM's Salesforce.com. However, anyone assuming that the open source variant is cheaper will be disappointed. A comparison made at the beginning of 2006 indicated that Salesforce.com, 5-user edition, was being offered at a lower price than the open source alternative from SugarCRM.

In addition to licensing income, almost all open sourcers receive income from services: training, consultancy, as well as patch and upgrade services. Zend, for example, sells services such as a membership to the platform and charges a unit price per processor per year.



### 3.3 The established order and the new wave get together

At present, commercial open source software has become an inseparable part of the modern IT landscape, in part due to unconditional support from large corporations such as IBM, Sun and HP. Notably, the two largest proprietary software companies, Microsoft and Oracle, are having some difficulty with this situation, but they will certainly not allow their problems to continue unabated. The Open Source Software Developer Lab organization (Linus Torvalds's home base, which has only existed since 2000), the enormous growth in the number of new open source software organizations since 1998, and the acceptance of open source software by IT service providers, hardware manufacturers and a number of large proprietary software producers have significantly increased the importance of the open source movement. As a consequence, a world without open source is no longer conceivable.

#### **Proprietary-software suppliers: the open source strategist**

That part of the IT establishment that collects a great deal of its revenue from the sale of software licenses, players like Microsoft and Oracle, views open source as a direct competitor. The predominant mindset in this group is adversarial: Linux versus Windows, MySQL versus Oracle, Compiere versus SAP. Nearly every proprietary product will sooner or later be confronted by an open source alternative meant to compete with established players – initially in the less expensive range. And although one attempt may be more successful than another, the established companies are also making their first strategic moves regarding open source, admittedly in a very cautious manner. Established players are accepting certain open source products. For example, Microsoft recognizes SugarCRM, which immediately raises the question of what this means for competition from open source in other areas where Microsoft is active. Notably, Oracle adopts the strategy of avowing its belief in Linux as a replacement for Windows while claiming not to see any threat in such database products as MySQL or PostgreSQL. The company's view, on the record, is that database software is simply beyond the capacity of the open source community. Direct attacks on open source (for example, in repeatedly claiming that the quality is poor or the risks too great) have little credibility in the eyes of the customer when the company offering this kind of argument embraces open source in other (non-competing) areas. In any event, it is extremely doubtful that customers would ever accept such a head-on strategy from Microsoft, particularly now that it is seeking rapprochement with the open source world.

## Proprietary hardware and services: the open source opportunist

Any company that does not have much of its own software is in a relatively simple position. In principle, it doesn't concern HP or Dell if open source or proprietary software is being operated on their hardware. They are happy if the software works well at a competitive price for the customer who is using their hardware. So if they see greater opportunities in promoting open source over other software, they have no qualms about wholeheartedly doing so. While open source strategists have to defend a position on the software front, hardware suppliers are free to take advantage of opportunities: they are open source opportunists. HP has an aggressive rip-and-replace strategy: UNIX and Solaris are being shown the door and, in their place, Linux (and server hardware from HP) is being introduced. They will also try to perform the same trick in other areas of the software stack (MySQL and JBoss are to follow). HP has limited revenue from proprietary software and, as a result, can afford to fully convert to open source, as can Dell.

Since opportunists also regularly form alliances with strategists, examining the activities of five traditional players (Microsoft, Oracle, IBM, HP and Sun) will illustrate the range of strategic and opportunistic positions. We will begin with the two software companies listed above, as they have an entirely different position in the field than the other players mentioned. They will be followed by IBM, which has substantial interests in proprietary hardware combined with some interest in proprietary software. Next will be HP, a hardware supplier that has much less to lose insofar as licensing is concerned. Finally, we will look at Sun Microsystems, which is prepared to take greater risks and to invest wholeheartedly in an open source future.

### 3.4 Microsoft: the favorite target of the “open source movement”

The European GigaWorld 2005 in Prague was opened by Forrester's founder and CEO, George Colony, with the assertion that Linux was the biggest threat to Microsoft's existence. But what does that mean for an organization that, according to many, has always only had the wind in its sails, and that clearly has been able to make the right moves at precisely the right times in order to survive financially or, better, to excel?

A leaked internal report has made it generally known that Microsoft was well aware of open source and the threat posed by it as early as 1998. And Microsoft was undoubtedly already aware of this development prior to then. The leaked report is known in certain circles as the so-called “Halloween Document I,”<sup>5</sup> a name derived from its publication around Halloween and perhaps from the Halloween horror films about an unstoppable psychopathic slasher. Microsoft has confirmed the authenticity

of the document, while commenting that it does not represent the company's official view. The document was written by Vinod Valloppillil and Josh Cohen. Twenty-one people worked on it at Microsoft, and there were fourteen revisions of the original text. Clearly, work on the document was thorough.

The Halloween documents reveal that at Microsoft in 1998, the year when open source was being conceived, some circles were deeply impressed by the software products that the open source community was able to deliver, as well as with the manner of collaboration involved. We cite the first Halloween document:

**From research by Microsoft into open source conducted in 1998**

- OSS poses a direct, short-term revenue and platform threat to Microsoft, particularly in the server space.
- Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long-term developer mindshare threat.
- Recent case studies provide very dramatic evidence ... that commercial quality can be achieved/exceeded by OSS projects.
- ...to understand how to compete against OSS, we must target a process rather than a company.
- OSS is long-term credible ... FUD [Fear, Uncertainty & Doubt] tactics cannot be used to combat it.
- Linux and other OSS advocates are making a progressively more credible argument that OSS software is at least as robust – if not more – than commercial alternatives. The Internet provides an ideal, high-visibility showcase for the OSS world.
- The ability of the OSS process to collect and harness the collective IQ of thousands of individuals across the Internet is simply amazing. More importantly, OSS evangelization scales with the size of the Internet much faster than our own evangelization efforts appear to scale.

What is interesting in the analysis is that Microsoft identifies the long-term risks of failing to make use of a larger community. The Halloween document speaks about a “mindshare threat,” the danger that Microsoft might be unable to attract and capitalize on the talents of bright minds because third parties are excluded from the closed Microsoft community of software developers.

Whether this document was responsible for the expansion of the Microsoft's Shared Source community strategy<sup>6</sup> is difficult to prove. Shared Source is, so to speak, the open source alternative for the proprietary products of Microsoft. Selected parties (customers) are given the opportunity to take part. Developers outside Microsoft can also start their own projects on GotDotNet (Got.net) or, at the invitation of Microsoft, may be

asked to participate in an in-house project. GotDotNet numbers over 8,000 workspaces (software projects and schemes). Six thousand people took part in its largest project (concerning the Microsoft reference architecture); the smallest projects consist of single-person schemes. Anyone starting his or her own project may choose from a number of licenses. In most cases, the choice is for an “as-is” license, which means that the developers are not responsible if the computer crashes. The projects taken over by Microsoft are placed under specific Microsoft licenses, which are therefore not open source licenses.

To understand the open source strategy of Microsoft, a distinction must be made between its community activities and its commercial efforts. Microsoft perhaps does little in the commercial sphere with open source, but it is certainly extremely active on the community side in attracting more people into its own production program and in making use of external knowledge. The licenses might therefore be different, but GotDotNet is a living community that, in many respects, has an open-source equivalent: SourceForge.net.

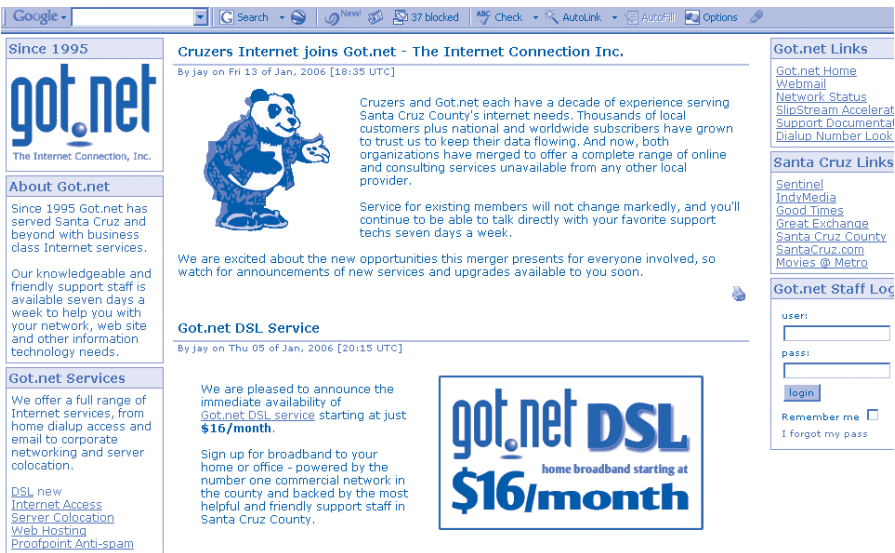


Figure 3.2: The software community helps Microsoft with a few thousand projects

Competing with Linux and other open source products

Linux had to be kept under Microsoft’s thumb, or so Microsoft thought. The plan was to cut it off at the pass, so to speak: “By extending these protocols and developing new protocols, we can deny OSS projects entry into the market.” When this strategy was revealed, it created an enor-

mous amount of bad blood in the old-fashioned peace-loving Silicon Valley community.

Officially, Microsoft described its respect for open source, demonstrated in the first Halloween document, as embracing the free exchange of ideas, which occurs in every healthy organization. Subsequent Microsoft annual reports give regular attention to open source as an ominous threat. Of course, this was part of the disclosure of comprehensive and truthful information to shareholders, but the suspicion remains that there was something more going on. Below is a brief selection of excerpts from the annual reports:

#### **Microsoft considers a potential shift in the market (annual reports 1999-2005)**

##### *1999: Possible fundamental transformations*

"The Company is faced with the possibility of paradigm shifts from PC-based applications to server-based applications or Web-based application hosting services, from proprietary software to open source software, and from PCs to Internet-based devices."

##### *2000: This development persists*

"The Company continues to face movements from PC-based applications to server-based applications or Web-based application hosting services, from proprietary software to open source software, and from PCs to Internet-based devices."

##### *2001: Competitive challenges*

"Open source software, new computing devices, new microprocessor architectures, the Internet, and Web-based computing models are among the competitive challenges the Company must meet."

##### *2002: The business model is in danger*

"To the extent the open source model gains increasing market acceptance, sales of the Company's products may decline, the Company may have to reduce the prices it charges for its products, and revenues and operating margins may consequently decline [...] the popularization of the open source movement continues to pose a significant challenge to the Company's business model, including recent efforts by proponents of the open source model to convince governments worldwide to mandate the use of open source software in their purchase and deployment of software products."

##### *2003: Open source is akin to theft and illegal copying*

"We face direct competition with firms adopting alternative business models to the commercial software model. [...] Additionally, global software piracy – the unlawful copying and distribution of our copyrighted software products – deprives us of large amounts of revenue on an annual basis."

*2004: Offense is the best defense, and 2002 concerns continue*

"While we believe our products provide customers with significant advantages in security and productivity, and generally have a lower total cost of ownership than open source software, the popularization of the non-commercial software model continues to pose a significant challenge to our business model, including recent efforts by proponents of open source software to convince governments worldwide to mandate the use of open source software in their purchase and deployment of software products."

*2005: Open source means that our earnings will fall*

"In recent years, a non-commercial software model has evolved that presents a growing challenge to the commercial software model. Under the non-commercial software model, open source software produced by loosely associated groups of unpaid programmers and made available for license to end users without charge is distributed by firms at nominal cost that earn revenue on complementary services and products, without having to bear the full costs of research and development for the open source software. To the extent open source software gains increasing market acceptance, sales of our products may decline, we may have to reduce the prices we charge for our products, and revenue and operating margins may consequently decline."

The forecast would seem to be for a drop in sales due to the effects of open source, but there is no sign of any such thing at this time. Business is still exceptionally good for Microsoft, although the company is concerned about what the government is going to do. It keeps a sharp eye on procurement policy, fearful that a guideline might be adopted under which the use of open source would become mandatory in the public sector.

Microsoft's labeling of open source as "non-commercial" disregards the fact that in 1998 the open source initiative was established precisely to drag "free software" (the name of the product group until that time) out of the non-commercial corner. In Microsoft's view, the average open source license returns too little income, and furthermore, quite a number of open source projects are "viral" in the sense that every added software code automatically enters the public domain.

### **From TCO to interoperability: Google becomes the greatest enemy**

The bond with the community of software developers is essential; selective parts of proprietary software are converted to open source. The strategy has shifted from an attack on open source to an affirmation of company strengths. Linux, the open source showpiece, hits Microsoft right in its product portfolio. As the most important exponent of the proprietary

software model, Microsoft is still the most successful IT company. Microsoft's answer to open source is, in theory, patents. Six years after the start of open source, Bill Gates flatly states that "increased, intense focus on protecting intellectual property" is Microsoft's answer to open source (*InformationWeek* 2004).

Microsoft's response to the increasingly more popular open source software has shifted over the years. For a long time, Microsoft drew market attention to business costs. This will continue, as costs clearly remain important in making a purchase. But in February 2005, Martin Taylor, Microsoft's general manager of platform strategy, explained that "total cost of ownership" will no longer be emphasized in discussing Linux. Taylor indicated that Microsoft will take a more practical tack following a study by Jupiter, an American think tank, which found that Microsoft was the best in terms of interoperability. Microsoft possesses a large number of XML-related technologies and patents. Companies are interested in a simple integration of applications, and XML is exceptionally well suited for this purpose.

Later in 2005, it turned out that this shift to webservices has everything to do with the new threat: Google. For that matter, warnings about the danger posed by this new rival were, in fact, already present in all the previous annual reports. Google and similar entities in the Web 2.0 realm are now seen as the greatest threat: the era of flexible internet is actually upon us. Recent moves by Google bringing YouTube, DoubleClick and Apple TV to the former search-engine-only company strengthen the concerns. With their internet collaboration and modular transparency, the more open-source related software "mash-up" business and the distributed manner of software development associated with it have made a significant contribution to this second coming of the Web.

Microsoft is a serious contender in the development of webservices and mash-ups. In December 2005, it published a mash-up of Dynamics 3.0 and MapPoint, Microsoft's online mapping service. With these mash-ups, users can link their customer data in the CRM system to a location diagram. At the start of 2006, more software extensions were released for applications including CRM and ERP. In connection with the Shared Source initiative, such activities occur under what is known as a Permissive License, which permits users to "view, modify and redistribute the source code for either commercial or non-commercial purposes." If that is not open source....

### **3.5 Oracle does not believe in open source business but still buys the companies**

Open source is a software development model and not a business model, Oracle claims. In other words, open source is an interesting way to produce software, but forget about making any money from it. Its pressure on the market is extensive, nevertheless. In April 2003 Larry Ellison, CEO



of Oracle, reiterated the company's position: "The database is the last piece of software that faces a threat from open source." In the meantime, Oracle has now acquired Innobase. Purchased in 2006, it is a Finnish company that maintains strong links with the open source community MySQL. In fact, Innobase supplies a storage engine for MySQL. MySQL also has a strong relationship with SAP.

Oracle's open source ambition is now growing expansively. In February 2006, *BusinessWeek* revealed that Oracle would like to buy at least three more open source companies, and wrote the following:

**Oracle can rival IBM for leadership of the open source movement**

"Overnight, Redwood Shores-based Oracle would rival IBM as the prime evangelist of a movement that's revolutionizing how software is developed and distributed."

*BusinessWeek*, 2006<sup>3</sup>

The biggest of the three fish that Oracle has been pursuing is the middleware company JBoss, a company that uses open source to compete with BEA systems, IBM and, to a lesser extent, with Oracle itself. With a total of 16 million internet downloads, JBoss estimates its own value at around US\$400 million. The second company on Oracle's wish list is Zend. The scripting language for websites that Zend provides (PHP) is used on 18 million websites. Zend is allegedly worth US\$200 million. Sleepycat is number three. It makes software used in numerous open source databases. In fact, Red Hat has just taken over JBoss for US\$350 million, but speculation about Oracle's open source maneuvers knows no end. A week after Red Hat signed the final contract with JBoss, Oracle boss Larry Ellison declared that Oracle would like to be a provider of the entire software stack, just like Microsoft. And this would include an operating system. Oracle has already been considering the acquisition of Novell, the largest Linux distributor after Red Hat. Perhaps shortly two flies will be swatted in one blow, as a takeover of Red Hat would incorporate both JBoss and the much coveted Linux operating system.

Oracle is well disposed toward open source as a development model. Undeniably IBM and Sun are better known for pro-open source stances, but Oracle's new acquisitive spirit might soon lead it to surpass both of them. Oracle has various irons in the fire insofar as open source strategy is concerned.

1. Oracle uses open source products (such as the Apache server) that it has further developed in the open source community in order to have their own products run well on them. The work done on these products is given back to the community. Oracle ensures that its own products operate well on Linux.

2. Clustering in open source. The Oracle clustered file system was developed in the open source community. Other database producers have, of course, also harvested the fruit of such efforts.
3. Oracle is actively involved with open source products associated with its own core products and has no fear of further collaboration with the community. It selects and implements projects for this purpose on the basis of business needs.
4. Oracle employees can choose whether to operate their laptops using Linux or Windows.
5. Some customers, such as Ministries of Defense, require the source code to be delivered along with the product. Oracle also has licenses that allow for this.
6. Oracle indemnifies customers, when they take delivery of their products, against possible legal risks associated with the open source components included in the products.

Oracle claims that it does not lose any sleep over discussions pitting Oracle against MySQL or other open source initiatives. The company does not perceive any real threat from the open source movement and even identifies some advantages. If a start-up begins operations with MySQL because its product requirements are not terribly stringent, the potential customer becomes used to working with databases. If such a company continues to grow, it will inevitably arrive at a point where the transition to Oracle becomes a serious consideration. Ultimately, considerations focus on features, functions and services. At a time when data integrity and real-time disclosure are high on the priority lists of many companies, Oracle faces the competition with confidence.

### **One open source competitor, Compiere, is an Oracle spin-off**

A striking fact is that ex-Oracle employee Jorg Janke is responsible for ensuring that open source competition has reached a higher level in the software stack. Janke is the leader and senior engineer at Compiere, an ERP product that industry analysts regard as one of the few serious open source ERP options. Whether he worked on open source development in his free time or in the boss's time as an Oracle employee is not part of the story. In any case, there is no restrictive policy against working on open source products at Oracle.

The Compiere product does make some demands on the customer. Specifically, the customized pieces must still be added and therefore programmed, which is, according to Oracle, an inconceivable step backwards in the development of ERP packages. Customers want one-step solutions. Undertaking ERP implementation with a manual in hand is an unthinkable step for Oracle's customers.

### 3.6 IBM is an open source evangelist with enormous closed source interests

IBM is a true evangelist and takes every opportunity to explain how much the company feels itself committed to open source and the community behind it. IBM produced an attractive commercial completely about Linux in which it was barely discernible that the commercial was by IBM. In its street advertising, IBM presents itself as one large penguin company.



Figure 3.3: Peace, love, penguin – and IBM, of course

#### A few of IBM's achievements

- In 1998, IBM began to integrate the Apache webserver into their WebSphere, which started the collaboration with the community.
- In 2000, Lou Gerstner announced an intention to invest US\$1 million in Linux.
- 650 IBM workers participate in the development community of Linux, Apache, Eclipse and the Globus project.
- IBM donated Eclipse, Cloudscape, voice recognition software and other software to the community.
- IBM released more than 500 patents in a so-called “pledge.”
- IBM purchased Gluecode and developed a new business model around free and open software.

The most talked-about fact in this list is undoubtedly the donation of 500 patents to the open source community. John Kelly, the senior vice president of technology and intellectual property at IBM, called this donation just “the beginning of a new era in how IBM will manage intellectual property,” but there is also another side to the story. Of a total 3,248 patents obtained by Big Blue in 2005, the overwhelming majority remain protected and, on balance, the company’s position unquestionably favors closed source. In fact, IBM has been leading the IT industry in terms of patent-list size for years. Still, open source specialist and Stanford professor Lawrence Lessig refers to the donation from IBM as a grand and substantial gesture.

The donation strategy of IBM invokes reactions from such people as Jonathan Schwartz, CEO of Sun. A few days after this splendid gesture from IBM, Schwartz accused IBM of old-fashioned proprietary policies. This accusation was made in an open letter to Sam Palmisano, CEO of IBM, in which Schwartz begins by announcing Sun’s plan to give its Solaris 10 operating system to the open source community.

“We’ve repeatedly passed along customer interest in having IBM support Solaris 10 with WebSphere, db2, Tivoli, Rational and MQseries products. Customers have made repeated calls to you and your staff. Those same customers have now asked me to begin communicating with you in a more public and visible way – they’d like the choice to run IBM products on Solaris 10, and they’re feeling that your withholding support is part of a vendor lock-in strategy.”

Jonathan Schwartz, 2005<sup>7</sup>

What Schwartz wants is support from IBM for Sun’s Solaris, and he draws support from customers who want the same thing. In Section 3.8, Sun’s strategy will be discussed further. For the moment, the letter is noteworthy for making it clear that players can have both open and closed strategies, perhaps even at the same time. The fact that IBM refuses to commit to supporting Solaris could, of course, have something to do with the investment in their hardware and software that would be required to get the system functioning properly. At the same time, IBM would undoubtedly like to close the door on this hardware producer who is, to put it simply, a competitor.

If the total software on offer is examined, IBM covers a great deal of the “stack” with proprietary software. But customers who would rather use open source alternatives might also be dealing with IBM, since it is happy to provide the service and support for this “competitor.”

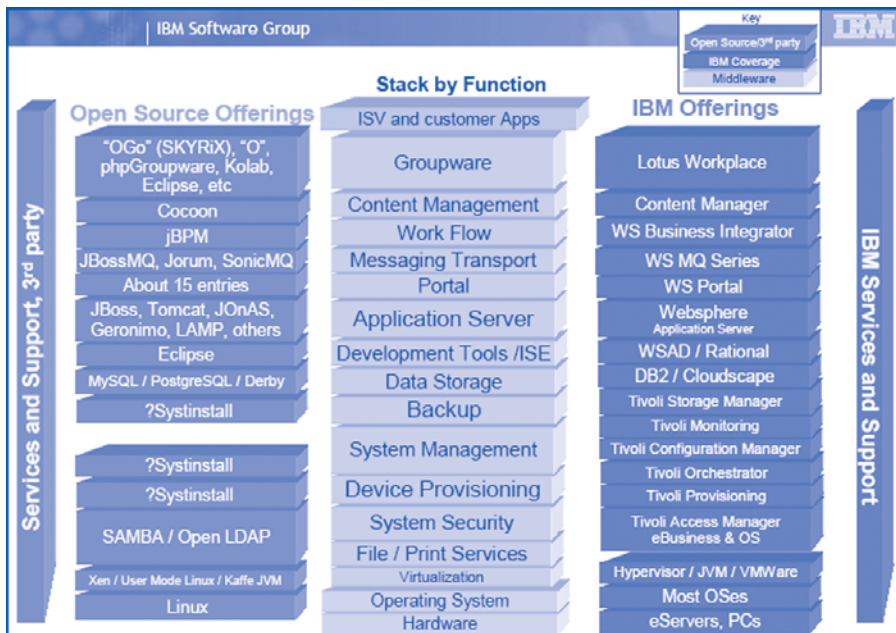


Figure 3.4: The products offered by IBM on the right; on the left, IBM's open source alternatives

### 3.7 HP specializes in hardware and services, and seizes open source opportunities

HP obtains the majority of its revenue from Linux-related hardware, and it is the market leader in the Linux server market (according to IDC). In fact, the company already has more than two hundred devices that run on Linux, and it is very active in the open source community. HP offers its customer an Open Source Reference architecture that is fully compatible with HP's Adaptive Enterprise strategy. Industry standards and a multi-OS strategy are the principal elements of this architecture. In practice, HP is especially cognizant of the trend whereby many customers of IBM AIX, Sun Solaris and HP-UX were migrating to Linux running on industry-standard servers (such as HP's ProLiant Blade server and Integrity server). In HP's Open Source Reference architecture, important building blocks consist of JBoss and MySQL, in addition to HP's OpenView and Serviceguard. HP offers customers indemnity for any potential legal risks that its open source products might incur.

Statistics indicate that HP is the largest Linux supplier, although it must be recognized that Linux involves third parties:

- Red Hat and Suse in the business market;
- Debian for embedded systems; and
- Mandrake for CAD environments.

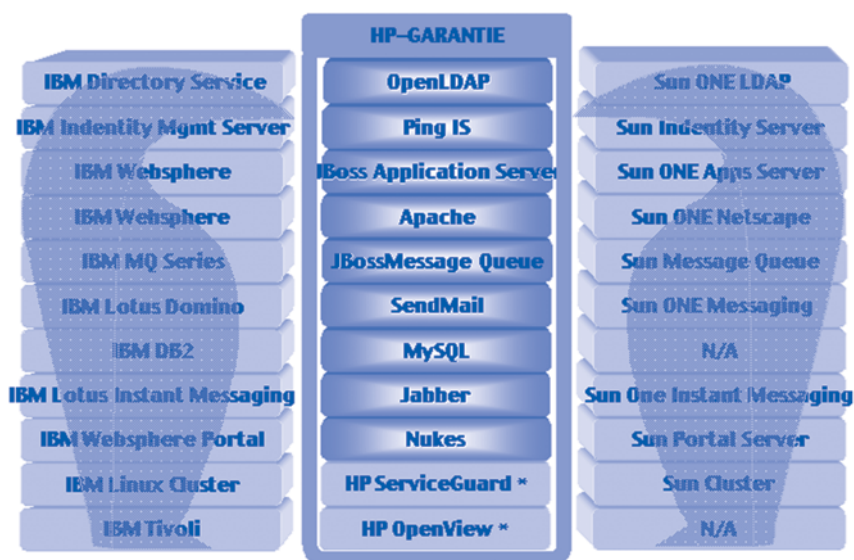


Figure 3.5: Open Source Reference architecture from HP

HP was the first to stand as a guarantor against the legal risks of open source; they offer indemnity in the event of loss. For the Red Hat commercial version, HP issues its own certificate of guarantee to indemnify customers against potential risks. As we have already seen, free software might come with a price tag if it includes risk coverage.

Anyone who thinks that everything from HP must be freely available is advised to briefly consult the 2004 annual report, which states:

“We rely upon patent, copyright, trademark and trade secret laws in the United States and similar laws in other countries, and agreements with our employees, customers, suppliers and other parties, to establish and maintain our intellectual property rights in technology and products used in our operations. Our revenue, cost of sales, and expenses may suffer if we cannot continue to license or enforce the intellectual property rights on which our business depends or if third parties assert that we violate their intellectual property rights.”

### 3.8 Sun produces its own hardware and software platforms, competes with Linux, and is pro-open source

Sun produces software and hardware but is much smaller than HP or IBM. The company made a startling move in open-sourcing its operating system.



### Now Solaris 10 is completely FREE!

Now you can use the Solaris 10 Operating System at home or at work – without paying a license fee. Download the Solaris 10 OS today – FREE – when you register your system (as part of the download or by going through the registration portion of the download) you will receive an Entitlement Document, which grants you unlimited rights to use Solaris-registered machines. Sun offers this new no-cost licensing program for all use – even commercial use, even on multi-processor machines. In addition to the free Solaris 10 OS, you'll also find our Java Enterprise System Software downloads available to you free to evaluate. Test drive this new and innovative product that is changing the face of the enterprise software industry.

The OpenSolaris community now numbers 11,000 members, of whom only 1,000 are on Sun's payroll. Evidently, this move could also be seen as a way of reducing development costs.

Why does a company give away a new operating system in which it has invested an estimated US\$500 million in recent years? That's not small change. This gift represents just under 5 percent of Sun's US\$11.2 billion revenue for the 2004 fiscal year. But of course there is an underlying reason for such generosity. Revenues continue to lag behind while Linux marches steadily on, although Sun explains its action in terms of the need to be more innovative. This link between open source and innovation is deeply rooted in corporate strategy. Ron Goldman and Richard Gabriel, two researchers at the Sun Microsystems Laboratories in California, were not acting out of the blue when they wrote the book *Innovation Happens Elsewhere: Open Source as Business Strategy*.



Figure 3.6: The book written by Ron Goldman and Richard Gabriel



In this book, the Sun researchers make an ardent plea for greater use of knowledge from outside of the company:

“It’s a plain fact: regardless of how smart, creative, and innovative your organization is, there are more smart, creative, and innovative people outside your organization than inside. Open source offers the possibility of bringing more innovation into your business by building a creative community that reaches beyond the barriers of the business. The key is developing a web-driven community where new types of collaboration and creativity can flourish.”

Ron Goldman and Richard Gabriel, 2005<sup>8</sup>

A press release of January 25, 2005, which explains why Solaris was open-sourced, emphasizes feedback from engineers. In making Solaris open source, Sun expects that engineers will gain deeper insight into the product and that new opportunities will arise for developers, customers and partners. “It’s the community, stupid!” is the position that Martin Fink adopts in his capacity as Linux VP at HP. It has certainly not been a decision without risk. One of the commentaries on Sun’s action was that the company should have done this much sooner. Why have they given Linux such a lead, only to have to play catch up? The reason lies in the fact that open-sourcing a piece of software requires a great deal of preparation. It took Sun seven years to do everything that was necessary.

### **Solaris is going open, but preparations took seven years**

1. First of all, the legal ins and outs were thoroughly examined. Making a product open source is one thing, but they had to be absolutely sure that all of the code being released was actually Sun property. Otherwise, Sun and its customers could become entangled in troublesome legal disputes.
2. Sun searched for a license that it believed was suitable for Solaris. The OS was issued under a Common Development and Distribution License (CDDL). This CDDL was approved by the Open Source Initiative (OSI) and, as a result, acquired the label of “official” open source.
3. A website was opened for the open source community, [www.opensolaris.org](http://www.opensolaris.org), which has now become the virtual domain where people inside and outside Sun can work on maintaining the code and devising innovations.
4. A governance structure for OpenSolaris was established in order to monitor the independent character of Solaris, its true open source character. The most important link in this structure is the Community Advisory Board, which consists of five members: two from departments at Sun, two from the newly made open source community and one from the broader open source movement.

In the media, Sun's move is seen as a direct attack on Linux, perhaps a slightly over-ambitious one. Sun claims superior product quality. Ultimately, Sun is trying to steal a piece of the pie from other processor platforms, especially the one involving Intel x86. Sun is essentially moving from a SPARC proprietary strategy to an x86 commodity strategy. As a second move, Sun has now also open-sourced the hardware from its proprietary processor. The UltraSPARC processor was placed under a GPL license in 2006. Sun is pulling out all the stops in order to expand its product base. The potential for Solaris to slow down the growth of Linux will, however, depend on the company's licensing policy. Those who select Linux will also automatically choose the General Public License, which also automatically makes all applications of the technology open source. Some companies want to avoid this viral effect. Now, anyone wanting to acquire a high-performance open source system but avoid this viral effect has a choice. The CDDL license from Sun will certainly be more to the taste of IT managers who fear the consequences of a GPL. However, the success of open source is ultimately determined by the contributions of the community. Sun's choice of a new license, the CDDL, is running into criticism from the community. The independent legal site, [www.groklaw.net](http://www.groklaw.net), has identified the main point of contention: the problem is not the fact that Sun has chosen its own license but the incompatibility of this license with the widely favored GPL:

"If Sun prefers to carve out a smaller community for itself, it is free to build its own little island, with its own big fence. The result will be, though, that Linux will continue to develop more quickly and it will bury Sun's license and its code, because the open, GPL method works better, and the GPL requirement of giving back all modifications results in rapid improvement. Sun is free to cut itself off from that, if it so chooses, but it will reap what it sows. If they imagined that the world would drop the GPL and adopt the CDDL instead, I trust by now they realize that isn't going to happen."

Groklaw, 2005<sup>9</sup>

The question as to whether Sun will have any success with Solaris outside its own familiar territory also depends on the support from other market parties, such as IBM. Linux is heavily supported by IBM, Novell, HP and Red Hat, companies that are ready to invest. At the same time, HP has a lot of reservations about Sun's move, which even motivated Martin Fink to tell the world his personal views by starting his own blog. IBM goes even further in announcing that it will not support its own software running on this new operating system. Returning to the open letter to Sam Palmisano of IBM cited earlier, Jonathan Schwartz insists that the transition from Solaris 9 to 10 is child's play.

Open source occupies a prominent place in Sun's strategy, more prominent than in the case of many competitors. All the same, we have to acknowledge that open source is not the only strategic card that Sun has to play, and not even the most important one. It is now all of twenty-three years since Sun prophetically exclaimed, "The network is the computer!" Everything is now in place to make this come true. Actually, Sun doesn't want to deal with customers about products, either hardware or software. Sun believes in services: utility computing. Accordingly, the issue is only what IT must do and for what price. Providing this variety of convenience has been represented on the Sun Grid since the spring of 2006, and it has been given the attractive and easily communicable price of US\$1 per CPU. Estimates are that the market pays a factor of 10 to 100 times more if companies undertake IT completely on their own, just as companies at one time all had their own power generation plants. The IT service variant is a very serious but also a heavy gamble by Sun. The market (that is, customers) is being asked to change its conduct, a shift that involves a lot of elements. Jonathan Schwartz says that the issue does not involve technology, as most of the problems are of a non-technological nature. And the one-dollar-per-CPU variant is a cultural change that could require much more time to catch on than was originally anticipated.

"We can try to make it safe; to make sure you can point out beacons or highlight customers that are saving a lot of money. But ultimately culture changes over generations. It doesn't necessarily change in an instant."

Jonathan Schwartz, *Forbes*, 2005<sup>10</sup>

Sun's open source strategy has two important cornerstones. One is Java and the other is its own UNIX operating system, Solaris. For the time being, Java is and will remain in Sun's possession, but there are persistent discussions about it, primarily sparked by ongoing pressure from IBM, which is constantly calling for its release.

### 3.9 Conclusions on open innovation

Reflecting on the first chapter, we see that IT suppliers are gradually getting back on their feet. The disruptive effect of open source is, as of now, not yet so dramatic as to bring the curtain down on "closed" companies. However, the curtain has indeed fallen on the closed business model.

Inspiration for a more concrete form of open innovation in other sectors can be drawn from what is taking place in the software market. If there is an inescapable point to make in concluding this chapter, it is that we are surprised every day by new collaborations and takeovers involving

traditional software players. The opportunism of hardware companies meant that they were the first to find the way to open source. But we would no longer be astonished if a software giant such as Oracle were to become one of the best young performers on the new open source stage.

The measures that such companies are taking make sense if we examine the interests that are at stake. “Follow the money” serves as a good guideline for understanding what is happening on the market and why the various parties are behaving in such a way.

This chapter contains numerous reasons for implementing a more open strategy. Here is an overview, intended to inspire and guide efforts at implementing open innovation elsewhere.

### **1 Not just ideas but cost reductions have a part to play**

Larry Augustin figures that open source could result in a 75 percent reduction in marketing and advertising costs and a 50 percent savings in R&D. The try-and-buy sales model of open source led to savings in the marketing budget, and the money saved on R&D was thanks to participation by the community in the company’s research and development. For example, Sun claims that a community of 11,000 individuals participates in the maintenance and innovation of Solaris, of whom only 1,000 work at Sun.

Although the try-and-buy model will not be an immediate option for many other sectors, it is certainly conceivable that marketing costs would decrease if an open source approach is adopted, as we saw in Chapter 2. This could be due to free publicity, like that generated by open source shoe manufacturer John Fluevog, or to clever internet marketing and word-of-mouth advertising in the community. The possibility that R&D costs outside the software sector could also be reduced is exemplified by what is already happening at Lego (see Chapter 2). Just under 6,000 new designs have been made by the community with only limited help now and then from Lego’s own R&D department. Without such cost benefits, Lego would not likely have been able to get the new business model off the ground.

Clearly, inspiration from open source can save a company money. Further benefits in adopting the production process of open source software will be explored in the following chapter.

### **2 Open is good for corporate image**

Suppliers rival each other in trying to be the “open source friendliest.” IBM even taped a peace sign, a heart and an enormous Linux penguin on the side of an apartment building. Open is good for corporate image. It reduces the disconnect between supplier and buyer. After all, users take part in the production and special licenses ensure that both producer and consumer can claim to be joint owners of the software. Software companies have realized that “open” is good for the image, and they play this

card against each other. IBM takes every opportunity to tell Sun that, if it wants to become a real open source company, it has to make Java open source. Sun writes in an open letter that IBM is not a truly open company, otherwise it would support the open source Solaris. And on and on. Suppliers want to become open source companies, and the positive effect on the company's image is certainly one of the reasons why.

Adopting open innovation strategies would enable companies outside the software sector to enhance their public images, too. Pharmaceutical companies could, for example, more actively support open research into medicines for tropical diseases. And more companies could make use of the broader community of technical innovators, as in the example of Boeing. It remains to be seen whether such practices must take the form of open source in order to generate positive public images. In the software sector, Microsoft is still on the defensive, at least partly because the licenses behind its community-source initiative are not truly open source licenses. However, the nature of these licenses is not so well known outside the software sector, and the establishment of a community (such as Boeing's World Design Team) may, in itself, be sufficient to create a positive spin.

### **3 Success primarily comes from the combination of internal and external knowledge**

Evidently, the success of open innovation rests on the combination of traditional closed business models with new variations of open innovation. Remember the table from Philips in Section 1.3 about the essential qualities of open innovation: "R&D by others can provide significant added value," and "We must profit from the use of our intellectual property and we must use the intellectual property of others befitting our business model." Business models in the software sector are seemingly becoming "both...and..." models: both innovating on the basis of R&D done in-house and making use of knowledge from external communities.

It goes without saying that companies are proud of their own discoveries, but "working" with others and being "open" have become full-fledged, money-making options. Returning to the table again, we've seen that, "Research does not have to be ours in order for us to profit from it." But software companies are even going so far as to willingly donate patents to the community. They do this because they believe it will ultimately enable them to obtain greater profits from patented items. In some cases you might even say, "Research need not be done by all of us in order that we may profit from it fully."

### **4 Open source strategies are adaptable to other sectors**

We began this chapter by listing seven distinctive open source strategies that producers use. It would be easy just to "copy – paste" these items into the sections of our discussion dealing with open innovation. In fact, this

step would be particularly easy for a few of the open source strategies. For instance, “consultation” about open source is one way to generate business, and consultation about open innovation is a strategy that could work in other sectors. Similarly, the optimization strategy could be applied outside the software industry: combining a company’s own products and ideas with those coming from the communities would increase value for the customer. Adopting such a strategy leads us to the essence of open innovation: “Not all the clever people work for us; we must find a way to tap into these other human resources.” The dangers of any resulting “mindshare threat” (as identified in the analysis conducted by Microsoft in 1998) can, in fact, be parried by implementing a more open innovation strategy.

What is difficult to translate into other sectors is offering two types of licenses (a dual-licensing strategy and internet-hosting strategy), subscriptions and the embedding strategy, which are rather software specific.

## **5 What happens if the competition does indeed begin to practice open innovation?**

If you think that open innovation is not relevant to you, consider what it could mean if your competition does, in fact, implement open innovation. Market newcomers appear with new business models and frequently enormous ambitions. If communities engaged in innovation develop around these new market players, there is still a question about whether or not an established company should wait to see which way the wind is blowing. In any case, an action-reaction has started in the software sector, as a result of which everyone has made the leap into open innovation.

Even the best-known players in the market may suddenly and unexpectedly change their game plan. We noted that Sun had open-sourced Solaris. Giving a large quantity of intellectual property “to the community” all at once can provide a jump start the consequences of which are still unknown. This type of action is to be expected precisely from those companies seeking better returns and stronger, more distinctive capacity (such as Sun in the software sector). And we also saw that producers like HP and IBM are acting as patrons. They support open-source communities, giving rise to a new interplay of competitive forces. Patronage will also certainly come into play in other sectors and provide support for communities. This will happen as players seek to share in the rewards of an improved public image, as mentioned earlier.

## **6 Open innovation can also be bought**

The patronage strategy used by companies like IBM and HP to support open source communities, has the advantage that it provides immediate intelligence as to what the community is doing. Following and learning from what is going on is certainly a motive behind such “benefaction.” An

even more drastic step would involve the purchase of a community. This is perhaps putting things a little too strongly, but the activity of a company such as Oracle could be described as an open source and community buy-in. Acquiring Innobase and possibly other open source companies can quickly establish a lead over the competition. This example provides new and rewarding insight into open innovation possibilities in other sectors. Instead of building up a community on one's own, it is possible to integrate into an already existing one.

### Notes to Chapter 3

1. [www.dmreview.com/article\\_sub.cfm?articleId=1040264](http://www.dmreview.com/article_sub.cfm?articleId=1040264).
2. *Open Source Turns Strategic*, November 2005, [www.cioinsight.com/article2/0,1397,1892140,00.asp?kc=ctrss02129tx1k0000534](http://www.cioinsight.com/article2/0,1397,1892140,00.asp?kc=ctrss02129tx1k0000534).
3. "Open Source's New Frontiers," *BusinessWeek*, February 6, 2006.
4. Internetnews.com, *Open Source a Driving Force for VoIP?*, 21/04/2005, [www.internetnews.com/xsp/article.php/3499536](http://www.internetnews.com/xsp/article.php/3499536).
5. The Halloween Documents, [www.catb.org/~esr/halloween](http://www.catb.org/~esr/halloween).
6. More information about Microsoft's shared-source program is available at [www.microsoft.com/resources/sharedsource/default.msp](http://www.microsoft.com/resources/sharedsource/default.msp).
7. "An Open Letter to Sam Palmisano, CEO IBM Corp.," *Jonathan Schwartz' blog*, January 21, 2005, [blogs.sun.com/roller/page/jonathan?entry=an\\_open\\_letter\\_to\\_sam1](http://blogs.sun.com/roller/page/jonathan?entry=an_open_letter_to_sam1).
8. Goldman, R. & R. Gabriel, *Innovation Happens Elsewhere: Open Source as Business Strategy*, 2005.
9. Groklaw, [www.groklaw.net/article.php?story=20050205022937327](http://www.groklaw.net/article.php?story=20050205022937327), February 2005.
10. Sanders, T., *Sun Chief Talks Up the Grid*, *Forbes.com*, February 4, 2005.





## 4 What motivates an open source community?

Up to this point, we have been specifically focusing on the business side of open source and open innovation. But these commercial successes depend on social factors, one of the most important of which is motivation. What inspires people to participate in the open source movement? This chapter explores this question using research from Berlin, Kiel, Maastricht, and Stanford universities. Cultural and organizational principles are the other elements in the social cluster that lay the groundwork for the success of open source. These two form the subject of Chapter 5. Following each chapter, we draw a number of conclusions applicable to open innovation.

### **Better performances due to “flow”**

“Flow” is crucial for the motivation of open source developers. It is a sort of intensely focused trance that participants in open source projects can fall into – like being “in the zone.” Flow occurs in various situations, but the conditions that create it are prevalent in open source projects. This is due to several reasons, including the fact that participants can select their own challenges, the goal is often clearer and the feedback unambiguous. The characteristic of a project with strong flow is that people work in a very concentrated manner, which enhances the potential for good performance. In other words, the conditions of open source production increase the potential for excellence.

### **They do it to belong somewhere**

Just like a soccer club, the open source community is an environment where like-minded people meet each other. The sense of belonging, and the sense of pride in participating in an open source community, goes even so far that the vast majority of the members view open source as a defining part of their identity. And just as in the case of a soccer club, not everyone is on the first team; backup players and coaches are just as important. For example, the more testers there are in an open source project, the earlier that bugs will be detected. The speed of development and the quality of the software product benefit as a result.

### **It is a part of a “gift economy”**

There are no invoices exchanged or payments made within an open source community. One performs a service for another without asking anything in return. At first, this might seem unique, but this “gift economy” behavior is present wherever people feel comfortably at home – it typically occurs in family circles. Outsiders may perceive open source developers as altruistic, but participants in open source projects mostly find that they get more from the community than they themselves put in.

### **It is not communism**

Bill Gates once compared open source to communism. However, anticapitalist motives underlie open source only to a limited extent, as most open sourcers certainly want to make money and only a small proportion proclaim any interest in subverting the power of large corporations. An extremely large proportion say that their primary interest is in promoting their own careers. Moreover, it is often people who seek to benefit from a particular open source solution who devote the greatest amount of work to a project. This symbiosis of community feeling and self interest has very little to do with communism.

## **4.1 Open source software does not and can never amount to much**

Before examining the question of what motivates the open source community to do what they do, it would put things in perspective to consider the actual results of these efforts. After all, what kind of quality can be expected of software created through informal and unpaid collaboration within loosely defined online communities? Conventional wisdom is that informal and unpaid labor can’t produce good products, and the ephemeral nature of an online community only increases the conviction that the products cannot be worthy.

Of course, open source software development takes place far outside the scope of the average product developer, seller, marketer, administrator and organizational board member. They certainly have better things to do than fuss about open source software. While compiling research for this book, the authors repeatedly heard IT managers say that they had previously regarded open source as more of a problem than a solution. It started as a loosely organized and unpaid online open source community, originally supported by very motivated experts preoccupied with standardization, infrastructural and architectural issues. But its impact can

hardly be overstated. Shouldn't this present a reason to take open source more seriously?

To illustrate, we will provide five examples of challenges that might not give the average commercial user of information systems pause for thought, but in software circles they are considered to be fundamental. Open source communities have played a crucial role in recent years in finding solutions to these challenges, solutions that have been subsequently adopted throughout the IT sector.

1. **Operating systems have become too complicated over time.** Repeatedly, modifications have been made and functionality added. The monolithic structure and all the updates have resulted in an inextricably entangled mass of spaghetti programming that threatens to sink under the weight of its own complexity, with regular system crashes being the inevitable but also unacceptable result. The neat solution propagated by the open source community is a clear structure of a well-defined kernel (microkernel) containing the basic functionality surrounded by driver modules offering additional functionality.
2. **The interests of commercial companies prevent the creation of a truly open platform that anyone can add to and improve.** But the success of the internet is essentially attributable to reliable, secure open-standard software, such as the webserver from the Apache open source project. It is unlikely that this success would have occurred if the infrastructure had remained in the hands of commercial organizations.
3. **In many situations, the excessive functionality of commercial software stands in the way of efficient and effective use.** In such cases, the limited functionality of an open source database such as MySQL, for example, provides an outstanding alternative. It is such a good alternative, in fact, that commercial software companies are now also making simple introductory versions of their software products available, some even for free.
4. **The transparency of the internet is impaired by web browsers that withdraw from the open standards of the World Wide Web Consortium (W3C).** Although the development of standards has fallen behind the desired rate of innovation, the value of standardization remains undiminished. The popularity of the Mozilla Firefox open source browser, which uses clear interfaces to interconnect with the underlying operating system and which is entirely based on open standards, constitutes a significant countervailing force to challenge commercial corporate interests.
5. **The re-use of programming code has, up to now, proved largely illusory, although putting such an ideal into practice would certainly turn the productivity of the IT world up a notch.** Open source programming environments such as the PHP and Perl scripting languages, as well as Eclipse (for WebSphere) and MONO (the open source counterpart to .NET), have enormously encouraged standardization and the prolifera-

tion of bits of programming code. Partly because of this, re-use has certainly taken off in open source communities.

The open source community has actually contributed solutions to such important challenges as complexity, application of open standards, excessive functionality, damage to openness due to commercial interests and the potential for re-use. These solutions are now generally recognized as “best practices” in the IT industry. The ceaseless efforts of a number of open software developers are responsible for these successes. They now receive strong support from large IT companies such as HP, IBM and Sun. Even organizations like Microsoft and Oracle, whose market position had once insulated them from concerns about open source development, now see the value in it.

We all know the importance of overcoming initial prejudices and deferring judgement until sufficient information is available. Evidently, the original objections concerning the dubious quality and continuity of open source software have proven unfounded. Further insight has shown that the quality and continuity are precisely some of the greatest achievements of the open source software community, although perhaps these achievements go overlooked because the community itself now takes them for granted.

## 4.2 Hacking open source and the passion of the individual

Open source developers are truly hard at work. Not only are they producing valuable software, but they also frequently demonstrate a capacity to come up with appropriate solutions to fundamental IT problems. This makes the question about what motivates the open source community to do what it does all the more intriguing. Various universities (Maastricht, Stanford and Kiel) conduct research inside the open source community. Working on open source projects, it seems, simply makes participants feel good. It produces a feeling of independence, and of doing meaningful, creative work. This colourful outburst by Migs Paraz on the internet paints a portrait of an average open source software devotee:

“I’m supposed to be a techie, but I’m not into games. I’m not into gadgets. And I’m not into tweaking hardware. Instead, I spend too much time playing with software – using, configuring, and coding. When I’m not hacking on work, I explore in P2P (Gnutella and BitTorrent), social networks (like blogs and Friendster), and other odds and ends that fascinate me. [...] I dream that one day, I can write open source software the whole day, just for self-fulfillment and without commercial pressure. To achieve that... let me get back to work.”

Migs Paraz, 2002<sup>1</sup>

Paraz fits the profile of a person who likes to play with his favorite toy. But games, gadgets and hardware do not appeal to him at all; experimenting with peer-to-peer software and with a social life online – that’s his thing. And it is his dream to have, at some point, enough time to completely devote himself to writing open source software. Paraz must certainly have earned enough money to do that. Paraz is a likeable “young researcher”: the prototype of the digital sorcerer’s apprentices who currently frequent open software forums in droves. In this way, Migs Paraz is far from the elite of open source gurus, such as Jason Hunter of the Apache team. At most, Paraz is an “active user.”

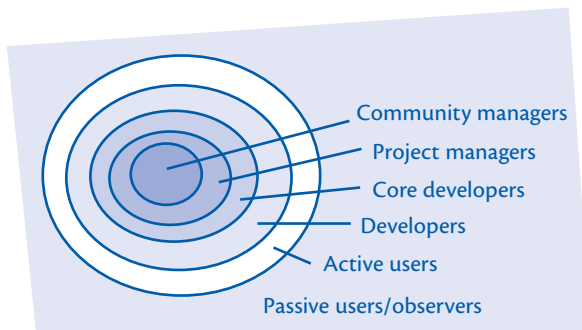


Figure 4.1: Various roles in the open source software community

The majority of the open source guys (females make up only 2 percent of the community) are strikingly motivated in their work and can be characterized as go-getters. They enjoy brainteasers and can search for bugs until they drop. Kazushige Goto is undoubtedly a good example of such a brain-twister type. He began by devising ideas for a faster computer while commuting to work, and ended up working at a university in Texas in order to convert his ideas into reality. Goto plunged into the quest for a better way to solve complex linear equations. His investigations resulted in his Basic Linear Algebra Subroutines: they allow computer processors to perform calculations faster. In fact, Goto taught computer architects a valuable lesson: he demonstrated that their designs could be profoundly optimized, and that it was high time to similarly optimize their hardware.

The story of Goto began in 1994 with the purchase of an Alpha processor workstation from Digital Equipment Corporation that he needed for his work at the time. The slowness of the machine surprised him. He investigated the problem and ultimately succeeded in obtaining 78 percent of the theoretical peak performance of the Alpha processor, instead of the original 44 percent, by using new subroutines. Goto found the necessary information required to achieve this result on a Linux-for-the-Alpha-chip mailing list. During his daily commute, he worked steadily on his new

vocation and often continued on it through the evening until midnight. Was this a punishment? Not at all: “I did it to relax.”<sup>2</sup>

Later, Goto was able to convert this special hobby into his work at a university in Austin, Texas, where he devoted himself to the Pentium 4 processor. He improved the performance from 1.5 billion to 2 billion calculations per second, out of a theoretical limit of 3 billion. Of course, Kazushige Goto is a unique phenomenon, but the fact remains that he obtained his knowledge and skills just from the internet. Then, as an amateur, he was able to make improvements that far exceeded the collective gains of the traditional “experts.” Kazushige Goto is currently working on an open source version of his Basic Linear Algebra Subroutines.

### 4.3 Who are these open source developers?

Statistics from Evans Data Corporation, an agency that specializes in research into trends in software development and that is active in more than seventy countries, indicates that there are three million open source software developers worldwide. They are spread across America, Europe, and the rest of the world. This is probably an accurate estimate. Considering the number of registered participants on internet sites such as SourceForge and CollabNet, it’s clear that millions are involved. SourceForge has over a million registered participants; CollabNet more than seven hundred thousand. Additionally, we need to recognize that the vast majority of the open source community does not program but reports bugs. There are fewer and fewer real core developers: estimates vary from ten thousand to thirty thousand individuals.

In trying to answer the question “Who are the open source developers?” researchers at the Technical University of Berlin interviewed over five thousand open source software developers in 2001. Large-scale research was also undertaken at the University of Maastricht and Stanford University. This research enables us to identify the key characteristics of open source software developers.

#### **Gender** (*TU Berlin, 2001*)

98% of the open source software developers are male  
2% are female

#### **Age** (*University of Maastricht, 2002*)

The average age of open source software developers is 27 years old  
More than half of them were active in the community before they were twenty

#### **Marital status** (*University of Maastricht, 2002*)

80% are single and childless



**Education** (*TU Berlin, 2001*)

60% have completed or are engaged in higher education

**Income** (*University of Maastricht, 2002*)

45% earn less than 2,000 euros per month

16.7% are still students

**Employment** (*TU Berlin, 2001*)

45% earn less than 2,000 euros per month

16.7% are still students

**Employment** (*TU Berlin, 2001*)

80% have IT-related jobs

20% of open source software developers are paid by a company for their open source work

**Work time** (*Stanford University, 2003*)

50% work on open source software during working hours

40% do this with the permission of the boss

**Open source software projects** (*University of Maastricht, 2002*)

On average, an open source software developers is involved in 1.5 projects at the same time. An average of 1 day per week is spent on it. If project participation is paid for by a company, over 2 days per week are spent on open source.

The research at Stanford University indicates that a large proportion of open source software developers are like Kazushige Goto. They crave the (technical) challenge of making improvements to what already exists, which involves figuring out how things work in the first place. Although there are a large number of reasons for active participation in an open source community, a fascination with technology and how to improve it is fundamental, which is illustrated in the following research into why developers take part in open source projects.

**Why did you choose to participate in [an open source] project?**

It seemed technically interesting	69%
I saw it as a way to become a better programmer	69%
I liked the challenge of fixing bugs and problems in existing software	40%
I wanted to find out more about how a particular program worked	57%

Stanford University<sup>3</sup>

## 4.4 Is there a taboo against making money?

The simple answer to this question is no, but that needs to be qualified in a couple of ways. The open source community is not adverse to money, but money plays a secondary role. Additionally, there is a small core of people (about 30 percent) who mainly see open source as a way of subverting the power of large IT companies. This may be characterized as a struggle against “big money,” but it is first of all a struggle for free and open software. In any event, Bill Gates regularly equates open source with communism. For Gates, the free and open source software movement is certainly anti-capitalist. The movement’s structure in the form of “communities” also invites comparison with communism. In any case, Microsoft identifies open source as “non-commercial” in its annual reports, as we mentioned in Chapter 3.

“Non-commercial” is certainly a subject of discussion in the open source software community (as well as outside of it). Nearly 80 percent of open source developers are of the opinion that proprietary developers are more interested in money than they are. But viewed in terms of a commercial enterprise, open source has noticeably outgrown non-commercial status, as was shown in the previous chapter.

The European Union suggests, in its turn, that the commercial efforts of large (American) IT players, such as IBM, Sun and HP, have misused the original open source community and that “our (*i.e.* European) open source software is a good and inexpensive option for ordinary users.” Jesus Villasante, Head of Software Technology in the Directorate-General for the Information Community, goes even further. He claims that the American multinationals have used/abused the open source community as a subcontractor. If IBM asks clients whether they want open source or proprietary software and their answer is open source, IBM then concludes, “OK, you want IBM’s open source.” Villasante is of course referring to the (understandable) opportunism of the large IT corporations. However, his accusing finger is pointing less at multinationals than it might seem. Villasante was specifically referring to the open source community when he said, at an open source conference in Amsterdam:

“Open-source communities need to take themselves seriously and realize they have made a contribution to themselves and society. From the moment they realize they are part of the evolution of society and try to influence it, we will be moving in the right direction.”

Jesus Villasante, *Holland Open*, June 1, 2005

Gates calls it communism, and Villasante says that open source developers allow themselves to be exploited. Research however indicates that only 30 percent of the open source community claims to work on open

source in order to limit the power of the large software companies. The reduction of open source to the struggle of communists against the established commercial order is therefore over-exaggerated. You could just as easily label the CIOs of companies as communists. After all, they have been attempting to curtail the power of the large software producers for years. The term “communism” is therefore not appropriately applied to this group of open source developers. If anything, the title of “freedom fighters” would be more appropriate. Still, nearly a third of the open source developers indicate that they participate because they want to increase their career opportunities. Their idealism is relative.

**Why do you work on open source?**

To improve career opportunities	30%
To make money	12%
To build up a reputation in the OS/FS community	12%

University of Maastricht<sup>2</sup>

Research at Stanford University<sup>3</sup> also reveals that “open” and “free” are both seen as important. Yet, as many as 80 percent of open source developers feel that software users have the right to view the code. A similar proportion of the community also believes that anyone should be able to modify the source code. To insist that, in principle, all software should be free would be to go a step further. Less than 40 percent are pushing for this more extreme position.

Open source licenses protect the principles of “free” and “open.” When asked, it turns out that nearly all open source developers regard these licenses as protection for the freedom of the software users. In addition, half of the developers feel that the licenses are important in order to obtain credit for their own work. As an extension of this point, 60 percent consider licenses to be a guarantee that others cannot appropriate their work.

**What are the roles of an open source/free software license?**

To prevent others from appropriating the software we've created	83%
To allow us to create OS/FS without scaring commercial firms away from using it	46%
To give credit to programmers' work	46%
To promote the launching of other OS/FS programs	43%
To protect the freedom that software users should have	60%

Stanford University<sup>3</sup>

## 4.5 Open source developers want to belong to the community

“Pleasurable feeling based on human relations generally makes man better; shared joy, pleasure taken together, heightens this feeling; it gives the individual security, makes him better-natured, dissolves distrust and envy.”

Friedrich Nietzsche, *Human, All Too Human*, 1878<sup>5</sup>

More than “being opposed to,” “belonging to” plays an important role in answering the question why open source developers do what they do. “Belonging to” and “being opposed to” are certainly related, but belonging and membership, as in a club, proves more important. Research has shown that group feeling becomes stronger as the pressure from outside increases. Therefore, the more that proprietary software producers attack open source, the stronger it could become.

“Wanting to belong to something” is actually too weak to express what is going on here. Taking part in the community is identified by open source developers as the most important source of their own identity.

The most important source of my identity is the open source software developer community

83%\*

I wanted to collaborate with like-minded programmers

57%

\* *strongly agree and somewhat agree*

University of Maastricht<sup>4</sup>

Before becoming involved in a selected group activity, one asks oneself a few questions. In a study conducted at the University of Kiel specifically aimed at the group activities of open source developers, this self interrogation boiled down to four questions determining the extent to which people would become involved in such a group.

Before participating in a group, people ask the following questions:

1. Do I believe in the force of this collective?
2. How will my environment react if I take part in the group?
3. Do I feel at home in this group?
4. Does it yield more than it costs?

University of Kiel<sup>6</sup>

First, consideration is given to the expectation of achieving the collective goal: is the open source community actually capable of making product X? Is it going to work? How likely is it that it will be a winner? And do I believe in the power of the collective? If the answers are positive, this is a factor in motivation.

A second factor concerns the response of the environment. Is it, for example, something that parents, friends and family would approve of? The more positive the reaction from the environment to the fact that a person is contributing to an open source community, the greater the motivation to participate.

Third, identification with the group plays an important role. An individual who more strongly identifies with a group is also more strongly motivated to do something for it. Anyone who fervently identifies with the Linux kernel group is intently motivated to do something for it, while the same person might only be moderately motivated to work on Open Office, for example.

Finally, the immediate payoff is also an important consideration: Will I get more out of it than I put in? Do I run any risks? Are the people I'm going to meet worth my time?

There is some debate as to whether a comparison between working in an open source community and working in teams is appropriate. A team consists of a relatively small group of people who work together, while an open source community operates as a network with hundreds and sometimes even thousands of participants. At the micro level, reliable teamwork certainly takes place in an open source community. The projects in an open source network mostly have a limited number of participants. Research at the University of Kiel again provides four fundamental explanations. In this case, it involves the question why are people inclined to work in virtual teams.<sup>6</sup> There are four factors:

Considerations before participating in a virtual team

1. To what extent am I attached to the team goals?
2. Can I in fact make any perceptible difference? Does it matter if I participate?
3. Am I effective in the team? Is my contribution good enough to bring about a result?
4. Are the people and the systems with which I will work reliable and trustworthy?

University of Kiel<sup>6</sup>

Working in virtual teams involves trust, belief in yourself and in the team, and raises the issue of whether your contribution would be meaningful. Notably, a study of the participants in the Linux kernel initially indicated that trust played hardly any role. Subsequent research revealed that trust was in fact essential, but that it has become unremarkable because it is present by definition in the open source production model.

It is interesting to see that a certain amount of self-assessment must precede participation in a virtual team. Asking whether you consider yourself good enough to make a contribution can lead to the realization

that you would be better off remaining on the sidelines. In this way, a natural mechanism arises to channel your energy into projects to which you feel best able to contribute. We especially want our contributions to be recognized; we want to play a meaningful role and to make a difference.

Open source developers are careful in choosing their projects. The commitment to the virtual team depends on an individual's ability to identify with the project goal. We are able to choose our projects and friends but not our families, and David Zeitlyn, an anthropologist at the University of Kent, compares open source communities to families. He finds parallels between the manner in which family members behave to each other when offering gifts and the way in which this happens in the open source community.<sup>7</sup> Basically, a "blood relationship" is a reason to do something for each other without asking ourselves what we might gain from it.

## 4.6 In this gift economy, it is a matter of give and take

Life is a matter of give and take, and an important part of open source software work is based upon this principle. You may have just got your hands on a bit of software for next to nothing and, consequently, you feel as if you've been assisted by strangers. And now you have the moral obligation to do the same for someone else. Nearly 80 percent of open source developers say that they want to do something in return for what they have received.

As a user of open source software, I wanted to give something back to the community 78%  
Stanford University<sup>3</sup>

A study by the University of Munich into the Linux community of embedded software reveals that the most important reason for donating code is that people regard it as perfectly normal to give something back to the community. The possibility that others will then elaborate the code and re-release it is said to be a second important reason.<sup>8</sup>

In fact, the gift is not entirely without self-interest, from the viewpoint of the open source software developers. Most have the impression that they profit more from the work of others than vice versa. In this way, they are getting a good deal, even though they do in fact give something back to the community. This is precisely the strength of the model: many hands make light work, and it is practically impossible for an individual to give more than all the others put together.

I take more than I give	56%
I give more than I take	9%
University of Maastricht <sup>4</sup>	

Giving yields something in return, although the reward is not necessarily any direct financial benefit and is possibly linked to a complex system of social collaboration. It is what the renowned French sociologist Pierre Bourdieu calls “symbolic capital”: the benefit that we receive in exchange for a gift.<sup>9</sup> Symbolic capital and real capital are two parts of the same economy, and are exchangeable. We can use one type of capital to acquire the other. For example, participants in an open source community might be hired by a company as consultants or could be offered salaried employment. In that case, the symbolic capital acquired by giving yields real capital in the form of money or a job.

But giving can also produce more direct returns. Much of the donated code is likely code that a person actually needs. Research at Stanford University<sup>3</sup> indicates that in eighty percent of the cases there was a specific need for the acquired solution.

<b>Why do you work on an open source project?</b>	
The software being developed would be useful to me	80%
I needed to perform tasks that could only be done with modified versions of existing software	56%
I needed to fix bugs in existing software	53%
Stanford University <sup>3</sup>	

Participation in open source projects evidently has a practical side. Indeed, the Apache user group functions for very practical reasons: people help each other mostly because it requires little effort and provides a bit of a distraction. It also provides useful insight into the issues of concern to users, and the cost in terms of time is very small.

Apache, which is used on two-thirds of web servers and is complex piece of software by any standard, does not provide any support for users. Consequently, an extremely active system for self-help by users has grown. They communicate with each other on Usenet. Apache-usenet is a success *because* Apache does not provide any official support. Half of the questions posed on Usenet are answered within a day, and satisfaction with the quality of response is high. What induces people to spend time solving the problems of others?

Karim Lakhani and Eric von Hippel conducted research into the motivations of participants on the Apache usenet.<sup>10</sup> They found that there were three important reasons for taking part, of which the most important was the desire to keep on top of problems affecting Apache. The pro-



portion of the time spent by these people in answering questions on the usenet was 2 percent. The remaining 98 percent of their time was spent reading the questions. Viewed in this way, the task of participating turns out to be a little less daunting than might originally appear. Individuals only answer questions in their field of expertise. No extra research needs to be done. Nevertheless, this still does not really explain why people participate in Apache user support. Lakhani and Von Hippel propose the following reasons, listed in order of decreasing importance:

- I have received assistance concerning an issue; I am now doing something in return.
- I provide answers in order to promote open software.
- I provide answers because I like to do so.
- I wish to strengthen my reputation in the community.
- I provide answers because I am taking a little break.

“I provide answers because I am an authority” and “I provide answers because it is a part of my job” scored the lowest of the options.

## 4.7 The gift economy is a family affair

The first step in an open source community is often a gift. You introduce yourself and make a suggestion about contributing something. Georg von Krogh, Sebastian Spaeth and Karim Lakhani undertook an extensive investigation of newcomer behavior in open source communities.<sup>11</sup> Below is one of the typical newcomer introductory scripts from the study. It shows that a newcomer may be uncertain about how best to contribute:

“I’ll be happy to look through the code and help out where needed, whether it’s heads-down coding, debugging, writing JavaDoc, or authoring whitepapers. Whatever. Until then, I’ll shut up and just absorb the culture a little bit and get my bearings!”

Georg von Krogh, Sebastian Spaeth and Karim Lakhani, 2003<sup>11</sup>

By offering gifts (software features), a newcomer can rapidly become known to the community. However, not all gifts are appreciated, or offered in answer to a clear need:

“I guess [developer #101] had some thread-pooling code that he’d previously written, which was just lying around, essentially. And he said, ‘Freenet needs thread pooling,’ so he just sort of imported that whole stuff in.”

The offered code was probably intended as an (easy) manner of making friends and connecting with the group. In practice, much work by open source developers does not end up in the definitive version of the software. Therefore, making and keeping open source friends is not always easy.

As mentioned, David Zeitlyn compares the culture of the open source community with that of close families. Because participants in the open source community are not really families, he uses the term “kinship amity” to describe the relationship involved in an open source group.<sup>7</sup> Here the mutual exchange of gifts is the central factor governing interpersonal relations. Other studies also make the link between gift economies and open source, such as the study by Jürgen Bitzer and Wolfram Schrettl at the University of Berlin.<sup>12</sup>

Family bonds are a product of your birth, but kinship amity is developed through interaction. You don’t know anyone when you first join an open source platform like SourceForge. Gifts have a crucial role to play in the process of interaction, in which you learn to know each other and gain one another’s trust. The generous exchange of gifts creates “kinship amity” and, in the open source movement, such gifts consist of software code that individuals have made themselves. Open source software developers build up relationships through interaction, build up trust by means of gifts and, in exchange, receive payment in the form of symbolic or sometimes real capital.

Economic relations among close family members are rather the exception than the rule. Normally, parents do not present their children with bills. What parents do for their children cannot be valued in real capital; they have given their children life, which means that children will always remain indebted to their parents. Conversely, parents also receive something in return from their children. In some cultures, parents depend on their children to provide for them in old age. On the other hand, there are also symbolic ways to repay obligation, ways involving attention or love. No bookkeeping entries or profit-and-loss accounts are drawn up on either side.

In a strong family relationship, everyone contributes whatever they can and receives whatever they need. Doing something for a family member is, in itself, its own reward. Bookkeeping is also absent within the open source community, although invoices may certainly be sent to the outside world. This possibly explains why Red Hat can still make money from providing Linux services, although “family members” do not hesitate for a moment to do what they can for each other pro bono.

## 4.8 The ultimate kick known as “flow” is the clincher

“Flow” is the experience of doing something so enjoyable so easily that you feel you’re floating on air. Flow is something experienced by painters, mountain climbers, writers, scientists, surgeons, factory workers, and programmers.

People who experience flow have one important thing in common: they become so immersed in what they are doing that they lose the sense of time. Everything seems to fall into place by itself. They enter into a kind of euphoric high. It is a kick that working on open source can also sometimes deliver.

Karim Lakhani and Robert Wolf certainly encountered this flow factor among the open source software developers they investigated in their research. Open source developers do, in fact, lose track of time when they are working, and can become so addicted to this sense of timelessness that they use every free hour to continue their work.

### Perspectives on Free and Open Source Software

I always or very often lose track of time when I am programming	73%
Participation in this open source project was my most creative experience, or at least as creative as anything else I have done up to now	61%
If I have a free hour, I “always” or “very often” use it in programming	60%

MIT Sloan School of Management<sup>13</sup>

Pleasure plays an enormous role. This has been demonstrated in a study by Martine Aalbers.<sup>14</sup> She investigated the motives driving the members of the Blender open source community, as well as others, to participate in an open source project. The pleasure factor scored the highest.

Mihaly Csikszentmihalyi is “the expert” on flow.<sup>15</sup> In his book, *Flow: The Psychology of the Optimal Experience*, he repeatedly describes the ultimate feeling of bliss experienced by various people in different situations. He defines flow in these terms:

“[Flow means] being completely involved in an activity for its own sake. The ego falls away. Time flies. Every action, movement, and thought follows inevitably from the previous one, like playing jazz. Your whole being is involved, and you’re using your skills to the utmost.”

Mihaly Csikszentmihalyi, 1993<sup>16</sup>

Csikszentmihalyi has been head of the Psychology Department at the University of Chicago for years. He has been researching the optimum experience that he calls “flow” for more than thirty years. He began his investigations in 1967 by asking how creativity actually occurred. At pres-

ent, Csikszentmihalyi is Professor of Psychology at the Peter F. Drucker Graduate School of Management at the University of Claremont.

The kick that we sometimes experience when doing things gives us, in a certain sense, a feeling of bliss. We forget the world around us and feel happy and content. In another sense, we are fully engaged. And it is in such moments that we deliver a top performance. Or, as Csikszentmihalyi says, “You’re using your skills to the utmost.”

## Flow makes work pleasant

Csikszentmihalyi makes it clear that flow occurs particularly in people who, in a sense, compete with themselves. In his book he describes a young man named Rico Medellin who worked on an assembly line. Assembly-line work is fairly boring, but Rico was able to make the work enjoyable for years. He made his task into an Olympic challenge by continually trying to break his personal record. He devised a special way to carry his tools and knew precisely which movements he had to make. He became celebrated by his boss and co-workers, but that was not so important to Rico. He did it primarily for himself, for the feeling of contentment “as if you were floating.” When he felt that he had reached the limit of his abilities, Rico began to take a few courses. Soon, he will begin a new task with other challenges.

Flow occurs when:

1. the activity is challenging to you;
2. it demands specific skills;
3. there is a clear goal that you want to achieve;
4. you receive unambiguous feedback during performance;
5. the task requires your full concentration
6. you feel that you have control over the situation; and
7. you are completely immersed in what you are doing.

After experiencing flow, you should become more self-assured.

Flow is an important factor in intrinsic motivation. The capacity to experience flow exists in everyone, and may occur when undertaking routine tasks as well as creative activities. In experiencing flow, it is as if we gain control over our own awareness. This awareness is necessary in order to perform, but it is not automatic. Csikszentmihalyi puts it as follows:

“In fact, it could be argued that chaos, not order, is the natural state of mind. When no external stimulation engages attention [...] thoughts begin to drift randomly. Instead of a pleasant, logical, thread of mental experiences, disconnected ideas appear out of nowhere.”

Mihaly Csikszentmihalyi, 1993<sup>16</sup>

## Flow stimulates the learning process

Flow not only makes work more enjoyable, it also enables us to become better at what we do. Flow constitutes the core of the learning process. This learning process takes place in the interplay between boredom and too great a challenge. Here we examine the example of Alex, and how a beginning tennis player deals with this tension (see Figure 4.2).

Alex is learning to play tennis. In the beginning (A1), he could do nothing at all. He concentrates on only one thing: getting the ball over the net. When Alex keeps at it for a while and things begin to improve, he experiences his first bit of tennis flow. However, the surge quickly ends as Alex becomes better. The following phase is boredom (A2). He will have to find a bigger challenge (for example, an opponent) to develop from A2 to A4 and experience flow again.

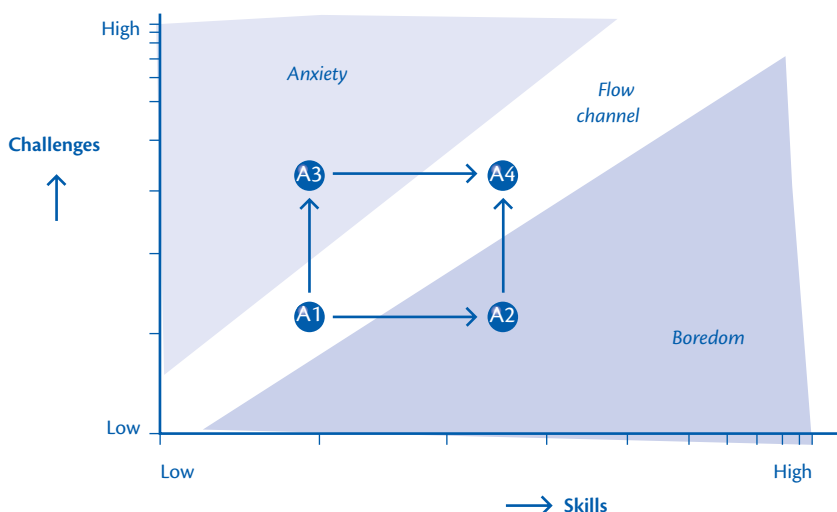


Figure 4.2: The flow Alex experienced

The other possibility is that Alex might encounter someone who plays tennis very well, and come to understand that playing tennis at a high level would be quite a task for him. He then enters into A3. This tension, up to and including straightforward fear (of failure), could cause Alex to quit tennis. There is only one thing to do: find an opponent against whom he can win. Alex then returns to a new state of bliss, but now at a higher skill level (A4).

The search for flow drives us steadily on to ever higher and more complex skill levels. We steer our thoughts and choose activities that, at the time, contribute to achieving flow.

The feeling of flow that we experience only gives us a glimpse into the quality of the skills involved. Someone could have produced a terrible

painting or a worthless bit of program code and still have experienced flow.

Both amateurs and professionals experience flow. Flow is linked to the frame of mind enabling us to concentrate to the utmost on a single task. It stimulates our capacity to “create” while balancing boredom and fear. This motivates us to learn new things, further enhancing our capacity to experience flow. In a study conducted at the University of Maastricht, the “desire to learn” was also identified as an important motivation in working on open source.

**For what reason(s) do you go on with developing and/or distributing OS/FS?**

To learn and develop new skills	71%
To share my knowledge and skills	67%
University of Maastricht <sup>4</sup>	

Karim Lakhani, the man behind the open source research performed at MIT, regards working on open source software primarily as accepting competition. In his view, open source is all about extreme competitive challenges. In a conversation with the authors of this book he expressed this idea as follows:

“Programming may be compared to driving a car. Many people can drive, but do not venture beyond normal traffic governed by its signs, stop lights and traffic police. Most programmers never do anything more than such everyday driving. Truly seasoned open source developers belong to another category. They are such good drivers that they like to challenge themselves in what could be described as Formula 1 races. Without any additional rules, they just drive as fast as they can and try to be the first to cross the finish line.”

Karim Lakhani, 2004

What drives the open source developer, as Lakhani concisely elucidates, is the same thing that drives Rico the factory worker and Alex the tennis player on to higher levels of performance. Taking on the challenge of competing with others or with yourself is what makes working on open source enjoyable. Driving yourself to the extreme without the limitations of rules and regulations is the essential thrill. At any particular level of performance, there is pleasure and an associated feeling of success, which is exactly what stimulates the learning process. As a result, learning and competition are often a good combination. Jason Hunster, one of the most important people behind the success of the renowned Apache open source server software, says that competition is essential in creating the will to work on open source software. In “Open Source from the Inside,” a presentation that he gave at the Palo Alto Research Center on October

31, 2004, he put it succinctly and directly: “Open source software development runs on extreme competition.”

## 4.9 Why open source is a rush

One of the most appealing features of open source is the fact that people can choose their own work. Imagine in your own workplace being allowed to ignore the unpleasant jobs and tackle only the things that you like. Likely your productivity would increase by leaps and bounds. In an open source community, this means that there are people who throw themselves into certain security provisions in the software, while others choose to do software testing and to report the bugs. According to Csikszentmihalyi, what people do for a living has nothing at all to do with what arouses their enthusiasm. This is why it is possible to speak of alienation and even a sense of wasted time:

“Many people feel that the time they spend at work is essentially wasted – they are alienated from it, and the psychic energy invested in the job does nothing to strengthen their self.”

Mihaly Csikszentmihalyi, 1990<sup>15</sup>

Every challenge can produce flow, but more is produced in an open source community – whether it focuses on software development or something else. The reason for this stems from the ability to choose one’s own work, which is not the case in most work environments. Mostly we are told what to do by someone higher up in the hierarchy. In software companies, this is a perpetual topic of discussion: do we have the right person doing the right task? Of course, every organization benefits from making the best possible match, for the sword is double-edged.

People who have other talents will seek out other ways, other hobbies, in order to experience flow. However, anyone who is handy with computer software can get off the couch, turn off the TV, and become the master of his own “bliss” by associating with a number of like-minded people somewhere in open source land. We can program something or test something, mostly as part of a large whole community divided up into smaller groups. If nothing suitable is found, you can think up your own challenge, a problem to work on, and start a new project. What’s important is finding a problem that challenges you to bring your knowledge and experience to a higher level – higher than what is required in the daily work that you do for your boss, for example. In any case, you are the one who has the say.

The intrinsic motivation called “flow” leads to extreme competition where the most important opponent is yourself. In open source projects,



the critical commentary from your fellow players is crystal clear. You communicate with code and receive code in exchange. Yours is flawed! This is better! The software does or does not work, and you are constantly attempting to improve your “high score.” A compliment makes you feel good, but once you have found a challenge with the “high score” that you wish to achieve, then the task is sufficient in itself. There is an elegant word to designate this state: “autotelic.”

## 4.10 Conclusion

Open source revolves around the intrinsic motivation of talented developers and around the insight into what drives the formation and functioning of communities. The lesson in this chapter is that working in communities results in increased motivation and heightened concentration on the task being performed. Participants work precisely on aspects where they believe themselves capable of making useful contributions. Self-regulation and motivation are strongly linked with each other.

Open sourcers feel a strong bond with their own community. We could even say that this has everything to do with self-fulfillment and lifestyle: the open source challenge in question becomes the most important element of personal identity. Every organization would love to have such a strongly bonded and committed community of workers. Generally, profit can best be gleaned from the engagement of a community if consideration is given to the following motivational factors, which have all been raised earlier in this chapter.

### **Eight decisive motivational factors**

1. I trust the (open source) environment.
2. I choose my own challenges and tasks.
3. I take part because I identify with the goals.
4. I work specifically on projects where my contribution is perceptible and I can make a difference.
5. I get back more than I put in, and I share my knowledge for this reason.
6. If I do things in a clever way, I can also promote my own career.
7. Working on open source lets me experience flow.
8. I can learn from the experience, both as a result of my own contribution and the reactions of others.

Trust is implicit and, after a person considers how best to contribute, participation can itself lead to such increased motivation that the person would prefer to spend every hour of the day working on the project. Ultimately, self-management by means of flow results in an ideal learning experience. This foreshadows our discussion in the following chapter on performing better in an open source community. Anyone who wants to

become more involved in an open source community should take this motivational lesson to heart.

In open source communities, high motivation certainly leads to better performance. Such a strong commitment is also desirable in other software communities. But companies are also forms of community. The essential difference lies in the ability of people to choose their own challenges instead of being compelled to work on something by their boss. We don't have to sketch an outline of a company to realize that self-management and motivation are present to a greater degree in open source communities. Self-management is perhaps easier to achieve outside company walls, but companies that seriously employ these principles internally can obtain great value: intensely motivated personnel who work concertedly to perform their tasks in the best possible manner.

Trusting people to do the right thing is not always easy. Companies working with an open community might think, "Since they do it for nothing and are not on the payroll, I cannot assign them any specific task." A (traditional) organization is also constantly in competition with other communities and lacks any effective coercive means of binding people to the organization. Companies therefore need to surrender control and place some trust in the community, or else the members of the community might up and move. This is one view.

Another view is based on the conviction that using trust as a model results in a better production method (this point is elaborated in the following chapter). This conviction is a pre-condition for the application of the open innovation principles internally in the organization. If this conviction does not exist, there is no reason for companies to do anything differently. After all, people on the payroll have a contract with the organization and are "obliged" to perform. Critics might say that employees also often leave organizations. Such cases frequently involve bright minds on the payroll who have been placed on the backburner. They are no longer getting the best out of themselves. Working in an open source manner can correct this under-use, as we saw in the example of Toyota in Chapter 2. We could then speak of open innovation inside the organization.

The fact that money is not a dirty word was demonstrated earlier in the examples of open innovation. In some communities, money stimulates flow. Money helps to motivate people even in open source communities. A part of the open source community is, in fact, in paid employment, and it is not unusual to use open source work to advance career prospects. However, there are still opportunities for organizations to unite their own objectives with those of the community. After all, it is a gift economy, and money can be an important part of the balance of give and take. However, the following three considerations are even more important:

- the drive that can be caused by "flow";
- the value that the participants obtain from participating in the community; and

- the realization of a new personal identity that makes life more meaningful.

#### Notes to Chapter 4

1. migs.paraz.com/w/archives/2003/11/23/a-few-of-my-favorite-things.
2. One Super Computer, *The New York Times*, 2005, [www.iht.com/articles/2005/11/28/business/geek.php](http://www.iht.com/articles/2005/11/28/business/geek.php).
3. David, P.A., A. Waterman & S. Arora, FLOSS-US, *The Free/Libre/Open Source Software Survey for 2003*, 2003, [www.stanford.edu/group/floss-us](http://www.stanford.edu/group/floss-us).
4. Ghosh, R.A., *Free/Libre Open Source Software: Survey and Study*, University of Maastricht 2002, [www.infonomics.nl/floss/report](http://www.infonomics.nl/floss/report).
5. [www.geocities.com/thenietschechannel/ma4.htm](http://www.geocities.com/thenietschechannel/ma4.htm).
6. Hertel, G., S. Niedner & S. Hermann, "Motivation of Software Developers in Open Source Projects: An Internet-Based Survey of Contributors to the Linux Kernel," University of Kiel, *Research Policy* 32, 2003, pp. 1159-1177.
7. Zeitlyn, D., "Gift Economies in the Development of Open Source Software: Anthropological Reflections," University of Kent, *Research Policy* 32, 2003, pp. 1287-1291, [opensource.mit.edu/papers/rp-zeitlyn.pdf](http://opensource.mit.edu/papers/rp-zeitlyn.pdf).
8. Henkel, J. & M. Tins, *Munich/MIT Survey: Development of Embedded Linux*, 2004, [opensource.mit.edu/papers/henkeltins.pdf](http://opensource.mit.edu/papers/henkeltins.pdf).
9. [nl.wikipedia.org/wiki/Pierre\\_Bourdieu](http://nl.wikipedia.org/wiki/Pierre_Bourdieu).
10. Lakhani, K. & E. von Hippel, "How Open Source Software Works: 'Free' User-to-User Assistance," *Research Policy* 32, 2003, pp. 923-943, [opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf](http://opensource.mit.edu/papers/lakhanivonhippelusersupport.pdf).
11. Krogh, G. von, S. Spaeth & K. Lakhani, *Community, Joining, and Specialization in Open Source Software Innovation: A Case Study*, MIT Sloan working paper 4413-03, 2003, [opensource.mit.edu/papers/rp-vonkroghspaeth-lakhani.pdf](http://opensource.mit.edu/papers/rp-vonkroghspaeth-lakhani.pdf).
12. Bitzer, J. & W. Schrettl, *The Economics of Intrinsic Motivation in Open Source Software Development*, Phillip J.H. Schroder Aarhus School of Business, 2004, [www.diw.de/english/produkte/veranstaltungen/earie2004/papers/docs/2004-301-Vo1.pdf](http://www.diw.de/english/produkte/veranstaltungen/earie2004/papers/docs/2004-301-Vo1.pdf).
13. Lakhani, K. & R.G. Wolf, *Why Hackers Open Source Software Developers Do What They Do: Understanding Motivation and Effort in Free/Software Projects*, MIT Sloan School of Management, 2005, [freesoftware.mit.edu/papers/lakhaniwolf.pdf](http://freesoftware.mit.edu/papers/lakhaniwolf.pdf).
14. Aalbers, M.E., *Motivatatie voor deelname aan een open source software community*, Vrije Universiteit Amsterdam, August 2004, [download.blender.org/documentation/bc2004/Martine\\_Aalbers/MartineAalbers.pdf](http://download.blender.org/documentation/bc2004/Martine_Aalbers/MartineAalbers.pdf).
15. Csikszentmihalyi, M. *Flow, the Psychology of Optimal Experience*, Harper Perennial, 1990.
16. Csikszentmihalyi, M. *The Evolving Self: A Psychology for the Third Millennium*, Harper Perennial, 1993; 2000, [www.dailyobjectivist.com/Heroes/MihalyCsikszentmihalyi.asp](http://www.dailyobjectivist.com/Heroes/MihalyCsikszentmihalyi.asp).



## 5 An open source culture is better

In this chapter, we will examine the emerging open source culture. When companies want to establish their own open source culture, they use terms like “community source” (Sun and IBM) or “shared source” (Microsoft). The fact that proprietary giants like Microsoft and IBM are also striving for an open source culture may seem remarkable, but on second thought, it’s perfectly logical. Open source does not promote anarchy, despite what some people think. Instead, it empowers experts who are quite capable of knowing how they need to work in order to achieve the best results. Therefore, an open source culture is a way to employ individual strengths, a high degree of self-determination and an organic form of collaboration within project teams in order to create a better product. We will conclude this chapter with a discussion of the similarities and differences between open source production, traditional software production and the “agile” intermediary form.

### Modular structure

Beginning to work in an open source manner means constructing modular forms of software. Otherwise, it would be difficult for programmers or testers to become involved in project development. This modularization or componentization has been an important issue in software development for half a century. In 2005, IBM announced that it expects to increase the rate of software development by 30 percent by implementing an open source culture.

### User-led and open innovation

Working in an open culture is the current trend in open innovation in the software sector. MIT Professor Eric von Hippel has been inspired by this development, discussing the notion of user-led innovation in his book *Democratizing Innovation*. Von Hippel promotes this idea to companies in other sectors, such as Philips, by means of his Lead User Concepts. The bottom-up, open nature of user-led innovation is primarily directed at “doing” and has achieved a long list of successes (from Apache and Linux to the community-source and shared-source initiatives of Sun, IBM and Microsoft), indicating that it is an effective means of countering over-regulation.

The traditional manner of software development is viewed these days as being overly managed. It leaves little room for bottom-up initiatives or self-determination. An open source culture corrects this. Open source is, at a minimum, an important and readily available supplement to traditional software production. Because individual users, software developers, large companies (including HP, IBM, Microsoft and Sun) and knowledge institutes (including the Software Engineering Institute and the Institute for Software Research) are permeated with it, we can rightly describe this as a culture of “open innovation.”

## **Our four exhibits**

Although the majority of open source projects do not get very far, a number are extremely successful. Open source development does not differ from “ordinary” software projects in this respect. To make the case for the innovative nature of open source, this chapter will sketch an image of it based on the following four elements: the research of Walt Scacchi, the views of Alistair Cockburn, the software development practices identified for decades with the Software Engineering Institute, and the initiatives to form an open source culture at IBM, Microsoft and Sun. Scacchi is director of the Institute for Software Research at the University of California. Alistair Cockburn is the most important representative of the “agile” concept typical of a new generation of software developers and closely related to open source.

## **Open source and agile programming**

Open source is primarily a culture of the virtual organization. There is a lot of bravura in an open source community, propagated by people who have earned their spurs in that community; this is a feature of meritocratic leadership. The ambience in which open sourcers work most closely resembles a playing field where players compete to win. Open source is a strongly competitive culture. Regardless of rank or position, anyone can look for assistance from those who think they know better. And although the manner of working may at first appear chaotic, closer inspection generally reveals an efficient system of allocating roles and responsibilities. Still, we mustn't stare blindly at open source without also examining other explicitly “free” and “open” types of production culture. Specifically, there are two systematically structured software development variants (*i.e.*, extreme and agile programming) that are related to largely unreplicated open source practices. This chapter will conclude with a comparison of the traditional manner of software production and the two new cultures: agile programming and open source.

## 5.1 Open source: a culture in which everyone seems to be doing something

“Culture or civilization is that complex whole which includes knowledge, belief, art, morals, law, custom and any other capabilities and habits acquired by man as a member of society.”

Edward Tylor, *Primitive Culture*, 1871<sup>1</sup>

Producing something in an open source manner means, above all, working in an open source culture. The culture or the customs of an open source society are derived from people who are strongly motivated to collaborate over the internet and do not mince words if they disagree about something. Open source software production works because participants give it their best shot. It is on this basis that people earn their authority. The culture of open source is therefore primarily focused on the encouragement of meritorious participant behavior.

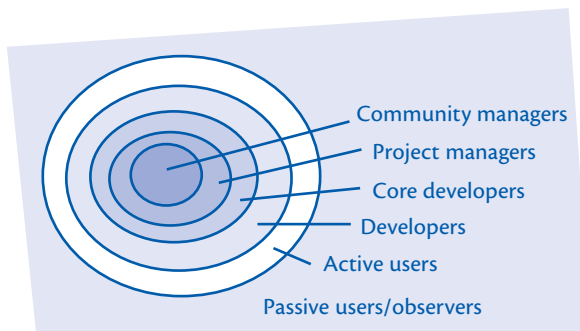


Figure 5.1: The proper role allocation for lean and mean production

Microsoft and IBM now view open source culture as something that is absolutely necessary for the effective development of software. If we take a good look at the current state of software development, we see that an open source culture and software development are more strongly inter-related than was originally thought. Classical, heavily controlled engineering culture has long since lost its prominence and is now frequently supplemented by cultures related to open source, such as extreme and agile programming. Even the professional software engineering world is being drawn in this direction, a trend shown in such articles as “Integrating Agile Practices into A Software Engineering Course.”<sup>2</sup>

We begin this chapter by specifically considering open source culture, and later we will shift our attention to its related offshoots. Together, they provide a good overview of where things are heading in the software development industry. In addition, we will discuss the essence of “open” and “free” production, along with the conviction – especially at IBM, Micro-



soft and Sun – that an open source culture is both feasible and necessary inside an organization.

The open source mode of production seems chaotic, or as Eric Raymond would say, it has a bazaar-like structure in which everyone has different agendas and everything seems to occur in a swarm of activity. This contrasts with the cathedral-building style of traditional software development, where everything appears to occur in a well-planned manner. The standard work on these contrasting production models, *The Cathedral and the Bazaar*, from 1997, is available on the internet in full text.

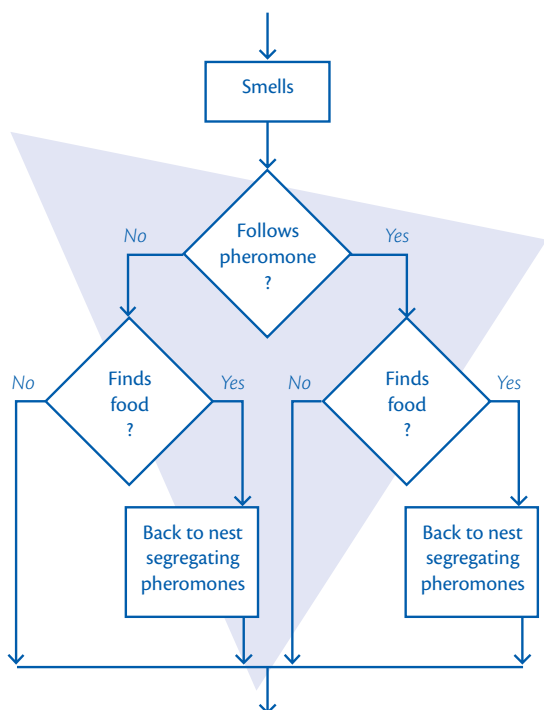
Curiously, the bazaar-like nature of leading open source development projects makes it possible to compare them to anthills. At first glance, we see all types of creatures running all over each other, while further examination reveals that everyone knows “by instinct” what their tasks are and how they are to be coordinated. Gregorio Robles, Juan Julian Merelo and Jesus M. Gonzales-Barahona published a study in 2005 that establishes an analogy between open source software production (which they refer to as “libre” software production) and an ant colony.<sup>3</sup> They make references to the work of the French biologist Pierre-Paul Grassé, who studied the ways in which termites build nests. Grassé suggests that they, in effect, begin by “just” getting on with the job. If a certain point in the construction is reached, others join in because they, in one way or another, feel compelled to do so. Grassé called this “stigmergy,” a term derived from the Greek words *stigma* (urge) and *ergon* (work). Figure 5.2 illustrates this form of self-organization.

One of the intriguing questions raised by such a self-management mechanism concerns the manner in which the social, technical and organizational production requirements are met: how do people know *what* needs to be done? Walt Scacchi has conducted extensive research on this issue.<sup>4</sup> He comes to the conclusion that open source does, in fact, have similar sets of requirements to other modes of production, but that you have to look very closely to detect them. They cannot be seen through a normal administrative-organizational lens.

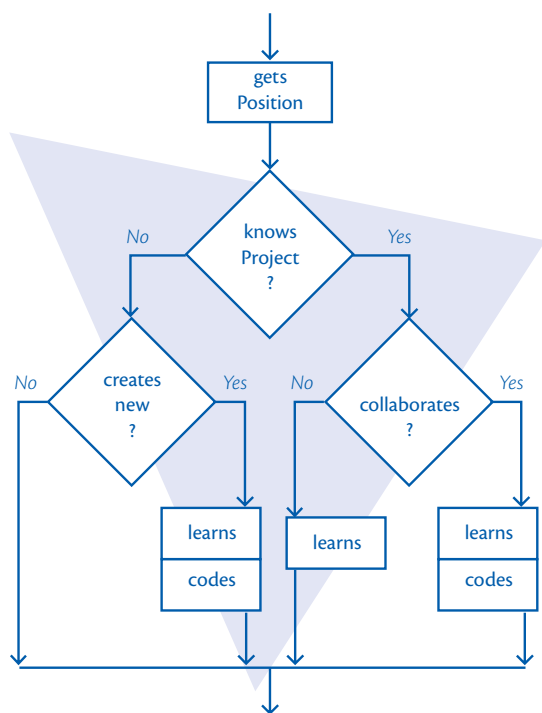
Instead, Scacchi claims that the requirements governing open source production emerge as part of a *community-building process*. It is a socio-technological process instigated by the development of social and constructive relationships. Informal social agreements are negotiated and commitments to participate in the production of open source software ensured. The composition of the community and supportive involvement in it are essential activities. Scacchi concludes:

“Thus, community building and sustaining participation are essential and recurring activities that enable open software requirements and system implementation to emerge and persist without central corporate authority.”

Walt Scacchi, 2005<sup>4</sup>



*Stigmergic algorithm used to find the optimal way to food from the nest*



*Stigmergic algorithm used to model libre software developers in each turn*

Figure 5.2: Building software like an anthill

Scacchi notes in passing that open source culture does not require any central authority. Nevertheless, there is certainly a hierarchy, along with clearly defined roles and procedures. We will come back to this point later.



Figure 53: Animal engineering: the ultimate smart structure<sup>5</sup>

Natural forms of stigmergy and self-management mechanisms are discernible in organically formed ant nests.

The organic bazaar culture in which work is performed has been described extensively by Eric Raymond. His influential book contains a long list of lessons and maxims.<sup>6</sup> We will mention nine of these lessons here, the ones that succinctly represent the self-organizational principle of open source software production. Raymond's self-management lessons can also be understood in the context of innovation in an "open" environment.

#### **Nine points for successful self-management**

1. Every good work of software starts by striking a nerve with an individual developer.
2. When you lose interest in a program, your last duty to it is to hand it over to a competent successor.

3. If you have the right attitude, interesting problems will find you.
4. Treating your users as co-developers is your easiest route to rapid code development and effective debugging.
5. Given a large enough base of beta-testers and co-developers, almost every problem will be characterized quickly and the fix will be obvious to someone.
6. If you treat your beta-testers as your most valuable resource, they will respond by becoming your most valuable resource.
7. The next best thing to having good ideas is recognizing good – and sometimes better – ideas from your users.
8. The most striking and innovative solutions often come from recognizing that your concept of the problem was wrong.
9. To solve an interesting problem, start by finding a problem that is interesting to you.

Raymond's tips indicate how self-direction can result in successful production. It begins with a personal response, which motivates people to take action and therefore to initiate production. There is also a "lock-in," since once you begin a program, you cannot drop it until you have found someone to take it over. The other points primarily concern cultivating the attitude required to collaborate well with others.

The organic growth that we find in larger open source projects (like the Linux kernel, Apache webserver or the PHP script language) is translatable into "organizations" with their own decision-making structures and principles. Some communities, like Debian and Apache, have formalized procedures to elect the people allowed to lead projects or modify the source code. Other communities are somewhat less democratic and are ruled by "benevolent" dictatorships, such as Linus Torvalds's leadership of the Linux kernel. The larger and more complex the project, the more important the procedures and management becomes.

## 5.2 "Merit" is crucial and users are involved

Open source communities are, in general, governed as meritocracies; the position of every individual is limited only by his or her talent and utility. Ethnicity, gender or wealth should, in principle, play no role.<sup>7</sup> In a meritocracy, a person climbs the social ladder by competing with others and beating them.

A meritocracy seems to be a more effective form of project management than traditional project organization.<sup>8</sup> Walt Scacchi shows that open source meritocracies sometimes display fully mature virtual forms of project management. For example, the open source community surrounding Planeshift, an internet game, has officers who are responsible for labor market policy (the recruitment of new open source employees), for issuing

press releases, for organizing events and for finding sponsors. There is someone who governs the rules of the game, another who controls the music on the site, and still another who manages the appearance of the virtual world.

User participation in open source software leads to particularly extensive feedback about experiences with the software, which subsequently results in improvements. In Chapter 4, we considered the factors motivating participants in open source development projects. Some participants were motivated by a personal demand for the improved software. The special quality of this user-driven innovation is its direct relationship with user experience. Consequently, we could speak of a “culture of experience” in these open source communities. Many so-called “bug fixers” are talkers rather than doers. They do not themselves “fix” anything in terms of making a contribution to the code, but they provide feedback or suggestions. Eric Raymond estimates that three-quarters of the total open source sector is comprised of this sort of member.<sup>9</sup> Extensive evaluation is a clear benefit in developing good software.

### **5.3 Towards an open source culture in the software development sector**

According to Sun and IBM in particular, the deliberate introduction of an open source culture within their company walls has not yet been given sufficient attention. Both view open source culture as a vital addition for increasing productivity in combination with the process-oriented software engineering and agile software development propagated in professional development circles. Agile software development is a concept specifically directed against the over-regulation that is characteristic of the traditional manner of software development. In traditional software development, also known as “software engineering,” programming seems to be pushed into the background. The open source recipe against over-regulation goes even further, inspiring MIT Professor Karim Lakhani to formulate the general research question, “Why do we need managers anyway?”

At present, the renowned Software Engineering Institute is also arguing for the addition of “agile” practices, which largely means what’s going on in the open source community. Enabling development professionals “just to do their work” appears to be the formula for success. Sun and IBM call it “community source,” the basis of which is modular system architecture. This is a “disentangled” structure in which the components are clearly linked together. Modular system architecture is the fundamental condition for working in an open source manner. The modularity allows a community to have a variable workforce. A strongly modular source code makes it easy for developers to work on it. They can focus on various

parts of the code without running the risk that a change or innovation will inexplicably paralyze the entire system as a result of some obscure entanglement of the program code. The community source approach leads to demonstrably fewer errors, as well as faster innovation and improvement of the program code.

The procedure has proven effective in the Mozilla project as well as others. In the case of the Mozilla webbrowser, a great deal of work was required in late 1998 to achieve the needed modularity, which is discussed in “Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code.”<sup>10</sup> The Mozilla program was so complex that it prevented those working in the open source community from doing anything with it. To put it bluntly, it had to be first transformed into community source, in this case by the open source community itself. Lotus Notes and the Longhorn/Windows Vista project are two important examples that show the benefits of adopting similar “community source” practices inside company walls.

### **IBM’s community source: making software 30 percent faster**

In June 2005, IBM announced that it would, from that moment on, be supplementing its traditional software development practices with community source: the official introduction of an open source culture throughout the entire organization.<sup>11</sup> The reasons for this move provide clear guidance as to how important open source is for software development and competitive strength.

The term “community source” is not an IBM invention; it had existed at Sun Microsystems for years.<sup>12</sup> At the end of 1999, Sun made it known, through spokesman Bill Joy, that it wanted to “community-source” as many of its products as possible.<sup>13</sup>

In June 2005 Doug Heintzman, Vice President Strategy and Technology at IBM’s Software Group, made it clear that, thanks to the purposeful linking of the pre-existing best practices (for example, the one involving the Rational Company that was integrated in 2003) with what he labeled an “open source culture,” IBM was able to develop its software as much as 30 percent faster.

The goal of IBM’s community source program is to promote transparency and simplification through modularization and componentization. In fact, this has been standard practice for software makers since the nineteen-seventies, but Heintzman claims that the cultivation of an open source culture makes the difference. As Heintzman describes it, IBM wants to combine strict rules and procedures with a bottom-up approach in order to tap into programmers’ potential for creativity and innovation:

“There is a very important role in a software company like IBM for top-down managed code architecture and all that kind of good stuff. But there’s also a tremendous amount of potential innovation that is locked up in the heads of the front-line programmers and we try to liberate that creativity and the innovative potential of all of those people.”

Doug Heintzmann, 2005

IBM has a division of “architects” given the task of examining how the desired modularization should occur and which systems should be given priority. As examples, Heintzman mentions IBM’s Lotus Notes and Microsoft Windows. The old Notes and Windows are good examples of the spaghetti code that poses immense engineering problems when making updates: there is simply no way of knowing the consequences a change will have on the rest of the software. For Lotus Notes, this problem has now been largely resolved by stringent componentization. In 2004, Microsoft also recognized a similar problem in what was formerly known as Longhorn, so they cleaned up the Windows code base and rebuilt it in modular form.

IBM itself has complete trust in this new open source RAD (Rapid Application Development), which has a component-based innovation capacity and should therefore speed up development by 30 percent. As we pointed out, however, these goals have been recognized as worthy since the seventies. In part, they have become more urgent due to the steady increase in software and system complexity over the years. We are continuing to chase the same goals in the hope that they may, sometime, truly be achieved.

## **Microsoft and the new architecture of Windows Vista**

Jim Allchin, the person responsible for the next version of Windows, belled the cat at Bill Gates’s house in July 2004. “This cannot go on,” said Allchin, referring to the fact that, with Longhorn, they crashed into the problem ceiling. How did that happen? Before the PC made its entry, programming was done in a methodical manner in order to ensure that the large computers at banks, in government agencies and on scientific projects would continue to function properly. In the eighties, a change took place. People began to develop software in a “quick and dirty” manner, as PC users were eager to pounce on useful new functions. Problems could always be solved later with a patch, or so it was argued. This practice functions well for a while, but there comes a moment when the complexity and intricacy of the code becomes so problematic that any further development is senseless. According to Allchin, Longhorn – which was the code name for Windows Vista at that time – had reached that point in the sum-



mer of 2004. There was nothing to do but choose a safe code base and work one step at a time rebuilding from this new foundation. Consequently, a historic decision was made on August 26 to “reset” Longhorn into a server version of Windows that was being developed for large companies. It was also decided to abandon the practice of having four thousand software engineers add something every evening to a new “build” of the system, after which testers would (often manually) go through the thousands of lines of code to resolve problems.

Two important measures were taken. First of all, programmers who produced too many bugs would be excluded. Second, Amitabh Srivastava together with Brian Valente had to monitor the modularization of the system. Srivastava did this by mapping how the Windows components were linked together on a chart measuring three meters by four meters. The objective was to make it always possible to add or eliminate parts of Windows without undermining the entire program. In addition, special tools were developed that the Microsoft Office group has also come to depend upon.

On July 27, 2005, Microsoft sent Windows Vista to a half million customers in order to have it tested. Instead of tens of thousands of bugs, the code turned out to only contain a few thousand problems. Things could still be better, but achieving this simplification and flexibility is already a giant step forward. As a result of improving the system architecture, a new build only takes a few days. Windows Vista remains an enormous Moloch, but it is at least a little easier to work on.

## 5.4 From software engineering to agile/open source software development

It is rather strange that it took so long to recognize that modular and transparent software systems were better, and for IBM and Microsoft to begin proclaiming the merits of an open source culture – the one explicitly and the other implicitly. Furthermore, closer examination reveals that “software engineering” was originally meant to be something entirely different from rationally managed and hierarchically organized software development projects. Alistair Cockburn, the most important representative of the now generally accepted “agile” thinking, has contributed a great deal to the systematic foundation of an open source culture. Cockburn’s ideas were first articulated in “The Cooperative Game Manifesto”<sup>14</sup> and coincide with the proclamation of open source, both dating, not coincidentally, from 1998. Both are strong arguments against the over-regulation of software engineering, which at the time was preventing programmers from getting down to their real work: writing computing software.

Cockburn's maxim states that software development is not engineering but ought to be seen as "Cooperative Games of Invention and Communication in Action." Making software involves realizing ideas in an economic context and can, therefore, be approached as a system of competitive and cooperative game situations. In "The End of Software Engineering and the Start of Economic-Cooperative Gaming,"<sup>15</sup> Cockburn provides support for this view. He is at his best on this subject in an interview recorded in July 2004.<sup>16</sup>

"In 1968 there was this NATO conference on software engineering. And when you read the preface to that [...] it says: 'We wanted to come up with a provocative term, and so we chose the term software engineering.' [...] In fact, what these people were saying is: 'We don't like the state that software development is, so we will throw out this word engineering [...] and see what goes from there.' [...] If you read through their description, they didn't understand what engineering was."

Alistair Cockburn, 2004<sup>16</sup>

Notably, Cockburn says that the word "engineering" was a sort of stopgap term. And although no one really understood what the word meant, exactly, it was nonetheless clear what had to happen to produce better software. A careful reading of the NATO text from 1968 reveals that, even then, the importance of bringing people into contact with each other was being proclaimed. The issues being discussed back then are all items that Cockburn would now include under the heading of "agile software development." Cockburn logically concludes there is essentially little difference between software engineering then and agile software development now.

"So, [...] there's a lot of similarity between software development and engineering [...] It's that they're both examples of cooperative games of invention and communication."

Alistair Cockburn, 2004<sup>16</sup>

In agile development, the emphasis is primarily on trust and empowerment – both within the development team and in the relationship with clients. Agile programming is primarily concerned with the activity of people and only secondarily with processes and tools. There is a strong collaboration with the intended user of the software; contractual negotiation is less important.

Later in the same interview, Cockburn made the following statements about what is truly important. They are all points he made in his Agile Manifesto.<sup>14</sup>

“We value:

- individuals and interactions over processes and tools;
- working software over comprehensive documentation;
- customer collaboration over contract negotiation;
- and responding to change over following a plan.”

In the latest book by Cockburn, there is a chapter entitled “Crystal Clear Applied: The Seven Properties of Running an Agile Project.”<sup>17</sup> It describes seven properties that the best development teams use to keep their projects running as smoothly as possible. Crystal Clear, designed for small development teams, is itself related to the first three. All properties, except for Osmotic Communication, are applicable to each project, regardless of its size.

- Property 1 Frequent Delivery
- Property 2 Reflective Improvement
- Property 3 Osmotic Communication
- Property 4 Personal Safety
- Property 5 Focus
- Property 6 Easy Access to Expert Users
- Property 7 Technical Environment with Automated Tests, Configuration Management, and Frequent Integration

In the previously cited interview from 2004,<sup>16</sup> Cockburn explains that Crystal Clear and agile development as such are primarily cultural, not procedural, concepts. You need to look for these seven properties and see if they are or are not present in a project. Cockburn says, “When I visit a project I always ask, ‘When was the last time you delivered any software? And when was the time before that?’ So the first property I look for is frequency of delivery. So frequent deliveries become the dominant property. And then I ask, ‘Do you ever reflect? Do you ever talk about what you’re doing and figure out how to get better?’” Cockburn’s second question concerns “reflective improvement.” He describes how asking questions and examining procedures can lead to embedding the desired agile culture. Personal security and trust are also crucial to software development.<sup>18</sup> Such trust can only be built up if people can freely admit when they don’t know the answer, make errors, or lack expertise. Such openness must be shared and fostered by everyone: other team members as well as yourself. This creates cohesion and trust in the group. That is why

- your knuckles are not rapped when you do not know the answer;
- you learn the “idiosyncrasies” of team members and no longer regard yourself as threatened, even when emotions run high; and
- you learn to resolve things together that you would not have been equal to on your own.

Agile development is certainly not a miracle drug, but it has proven its value in complex projects where many changes are involved. According to Siemens's Roman Pichler, experience teaches us that agile methods can increase productivity of software development teams by a factor of 2 to as much as 30. At Siemens, productivity was increased fourfold after one year.

In the next-to-last section of this chapter we offer a comparison of traditional, agile and open source software development.

## 5.5 Software engineering, components and the human dimension

The expression “software engineering” was officially used for the first time in October 1968 at a conference in Garmisch-Partenkirchen in Germany. In the previous section, we noted that Alistair Cockburn regards the introduction of this term as a stopgap measure that has misdirected a lot of people. Although the word choice might have been unfortunate, Cockburn still praised the conference for its shrewd analysis and recommendations. One of the proposals involved the introduction of a “components industry.” Dividing software into components is an obvious structural way to produce reliable software applications by means of standardization and re-use. We have known this for forty years, but are only now beginning to have anything to show for it.

“My thesis is that the software industry is weakly founded, in part because of the absence of a software components industry. [...] A components industry could be immensely successful.”

Douglas McIlroy, *Mass Produced Software Components*, 1968

As early as 1987, just twenty years after the above-mentioned NATO conference on software engineering, Frederick Brooks wrote that enormous efforts were necessary in order to make small advances in software and system development. Brooks, together with Gene Amdahl and Gerrit Blaauw, had formed the architect team for the IBM system/360. He described the problem by making an analogy with the emergence of modern medical research:<sup>19</sup>

“The first step toward the management of disease was replacement of demon theories and humours theories by the germ theory. That very step, the beginning of hope, in itself dashed all hopes of magical solutions. It told workers that progress would be made stepwise, at great effort, and that a persistent, unremitting care would have to be paid to a discipline of cleanliness. So it is with software engineering today.”

Frederick Brooks, 1987

In “No Silver Bullet,” Brooks warned that, insofar as software development was concerned, there is no hope of a technological or management change capable of simplifying or improving productivity or reliability like the changes we’ve seen in electronics and computer hardware.

## Toward the human dimension of agile software development

Software development is still plagued by projects exceeding budgets and deadlines, which often leave out the desired functionality as well. However, we are extremely busy disproving the “point” made by Frederick Brooks. We now know that a large number of problems are attributable to the exclusive use of conventional methods and practices in the development process. It has become clear that traditional approaches are not capable of overcoming key challenges, specifically those of a business and technical nature that arise in the middle of projects.

Traditional waterfall software-development methods were designed to avoid chaotic “code & fix” situations. Consequently, they established a disciplined process geared toward making programming more efficient and results more predictable. The understandable ambition was to escape undesirable upheaval. The strong emphasis on planning was inspired by the physical engineering disciplines. But too great an emphasis on planning is clearly counterproductive when change and adaptation is the norm. In that case, it is better to keep everything as simple as possible. It is frequently argued that change has become a given, because organizations and their processes are now “chaordic” in nature: ostensibly structured but actually unstable in reality. Agile software development will better suit these changes than the traditional “Taylorian” waterfall techniques, in which the process is linear and sequential.

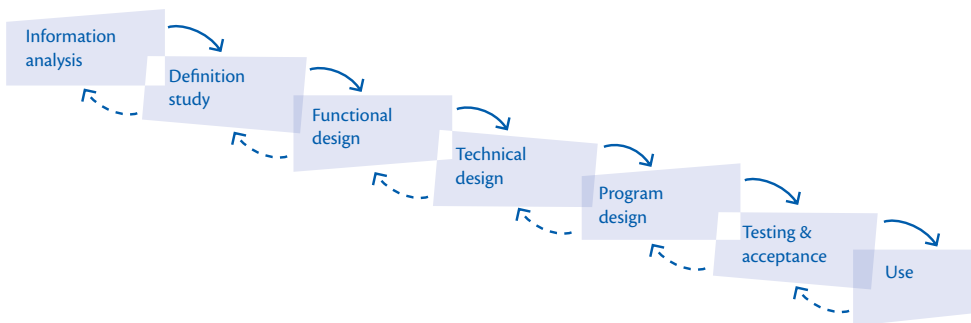


Figure 5.4: The traditional waterfall was an initial methodological paradigm. It has persisted, in recent decades generating such schemes as Evolutionary Prototyping and Evolutionary Delivery. A good survey of this trend is provided by the article, “Managing the Development of Large Software Systems.”<sup>20</sup>

Among other things, agile development prioritizes the most important software deliverables; work is done in short cyclical iterations (time boxes of three weeks or less) and tests are conducted continuously. Agile software development must be seen as the virtual variant of “lean production” in the physical engineering realm, which has a long history in Japan, especially at Toyota.

## 5.6 Open source software development in companies

A very real warning has often been issued that a company starting an open source project has no reason to expect that it will achieve results comparable to Apache or Linux. After all, is any in-house project attractive enough for developers in the open source community? Conversely, the introduction of an open source culture to a company’s software development organization and/or among the company’s partners is achievable in any organization that possesses a well-trained software development community.

Forrester Research’s best practices report, “Applying Open Source Processes in Corporate Development Organizations,” provides a finely qualified answer to the question that many software development organizations ask themselves: “How can we use the power of open source to improve the software development practice in our company in terms of output functionality, quality, costs, coherence, *etc.*?” Forrester’s answer is built from the following ten components:

1. Development processes can be improved bottom-up. Most organizations approach process improvement by implementing top-down initiatives, but the consequence is that these do not capture the hearts and minds of developers.
2. Organizational development can learn from open source:
  - Adopt a number of open source processes
  - Adopt a number of agile processes selected by the open source community
  - Disregard a number of open source processes
3. Adopt these open source principles:
  - team communication
  - allowing users to participate
  - the participation model (developers having different roles in several projects)
4. Build a robust structure for team communication:
  - Make documentation less of an obligation (use blogs, wikis, mailing-lists, CollabNet, *etc.*)
  - Make information a valuable asset
  - Increase transparency

5. Ensure that users are involved in development:
  - Reveal development work to internal users
  - Recruit a group of beta-testers
6. Staff projects flexibly:
  - Provide project workers with a certain degree of self-determination (so they can make their own choices)
  - Form teams of core and non-core developers
  - Raise the status of the people who supply the software code and pay them accordingly
  - Correspondingly reduce the emphasis on administrative tasks
7. Adopt agile processes that have been proven in open source projects:
  - Release software early and often
  - Formally involve users in development
  - Enact collective code ownership
  - Practice continuous code integration and employ automated testing processes
8. Open source follows rules better than corporate IT does:
  - Enforce peer code review
  - Construct a minimal code base and add new functionality in separate modules
9. Avoid these open source processes:
  - That developers chart their own courses
  - That developers derive all their motivation from intrinsic factors such as “recognition.”
10. The final corporate recommendations:
  - Automate parts of the documentation process
  - Get users involved in development as much as possible
  - Involve people in multiple projects
  - Explore agile techniques that have been proven to scale effectively

Forrester’s astute message is that top-down practices might have been necessary in the past in order to improve software development in organizations, but this has not been successful with programmers themselves. The announcement by IBM that it’s moving toward the community source concept underlines this fact, and such a move is, in effect, precisely what Forrester Research promotes through its analyses and recommendations.

However, such practices are much less available to non-IT companies, although they may develop a great deal of software and systems. For these companies, Forrester warns against the two “points to avoid” in item 9:

1. **Do not allow “too many irons in the fire at the same time.”** Although open source developers like to choose their own challenges, it is almost impossible to permit such self-selection in an IT department. Time and budgetary restrictions make such flexible project staffing impossible.



2. **Do not rely too much on the possibilities of intrinsically motivating developers in a conventional IT department.** What IT people do is often so invisible to end users that it cannot trigger effective intrinsic motivation. Nevertheless, it is certainly worth making some effort. It is undeniable that cultivating intrinsic motivation in those performing enormously complex tasks, such as developing software, represents the most effective means of improving performance.

Forrester therefore argues for a more bottom-up approach, as well as cautionary awareness of the points to avoid. How extensive any associated problems are depends upon the creativity of the organization and the “agility” of the software development, while the architectonic transparency is a necessary condition for the collaboration of various individuals in working on complex source code: “Construct a minimal code base and add new functionality in separate modules.” Clearly, “proven” agility and streamlined architectural components (from Cockburn to IBM) – in addition to other more common-sense considerations – have now become essential elements. In short, an open source culture is literally something for everyone. Provided that it is applied in a balanced and well-considered manner, this practical form of agile software development can serve companies well.

5.7 Differences among traditional, open and agile software development

Forrester makes a distinction between software engineering, agile software development, and open source software development (see Table 5.1). The boundaries between these three are certainly fluid, especially nowadays. Moreover, it might be preferable to call the agile method “community source” because it involves the application of principles from the whole culture of open source. Still, Forrester’s clarity is certainly useful in quickly making distinctions for the benefit of business and IT administrators.

	Traditional	Agile	Open source
Documentation	Documentation is emphasized as a means of quality control and as a management tool.	Documentation is de-emphasized.	All development artefacts are globally available, including code and information documentation.

	Traditional	Agile	Open source
Requirements	Business analysts translate users' needs into software requirements.	Users are part of the team.	The developers typically are the users.
Staffing model	Developers are assigned to a single project.	Developers are assigned to a single project.	Developers typically work on multiple projects at different levels of involvement.
Peer review	Peer review is widely accepted but rarely practiced.	Pair programming institutionalizes some peer review.	Peer review is a necessity and is practiced almost universally.
Release schedules	Large numbers of requirements bundled into fewer, infrequent releases.	Release early, release often.	Hierarchy of release types: "nightly," "development," and "stable."
Management	Teams are managed from above.	Teams are self-organized.	Individual contributors set their own paths.
Testing	Testing is handled by Q&A staff, following development activities.	Testing is part of development.	Testing and Q&A can be performed by all developers.
Distribution of work	Different parts of the codebase are assigned to different people.	Anyone can modify any part of the codebase.	Anyone can modify any part of the codebase, but only committers can make changes official.

Table 5.1: Forrester Research, 2004<sup>21</sup>

Placing agile programming in the middle suggests that it occupies an intermediary position between “traditional” and “open source.” However, “agile” is more closely related to “open source” than to the caricature of traditional software development presented by Forrester.

The principles of self-organization and peer review, the practice of involving users, and the rapid and frequent release of software are to be found in both agile and open source. They share a culture that we can label in Alistair Cockburn’s terms as “cooperative gaming.” Criticism is a standard feature in this culture, and it’s sometimes even necessary in order to enable people to learn about each other. Insiders in software departments

often reproachfully distinguish open source from agile on the basis of the uncritical nature of the former; open source is much less of a reflective activity, much less is described, which is precisely the unique character of open source development – and how it must remain. Further research into open source development could reduce this polarization. For example, Walt Scacchi has already shown in his research that open source has the same requirements as traditional development but that they are interwoven in the entire process of “community building.”

“Traditional” and “open source” lie at opposite ends of the spectrum. This is because traditional is, in fact, very precisely elaborated and described, while open source has, so to speak, fewer ambitions in this respect. The question whether open source is better than traditional remains unanswered, a point that was made in the introduction to this chapter. Instead of answering this question, it is much more productive to look at what is happening in reality. Open and agile cultures are on the increase, and they compensate for the failings of traditional methods, which are excessively riddled with over-regulation. The signals are clear. In the future, both extremes will continue to converge, and a software practice and culture that balances traditional, agile and open source approaches will emerge in many organizations.

## 5.8 Conclusion

Better performance in the software sector as a result of open source culture has, in fact, a lot to do with the nature of software production. In the examples we’ve seen from other sectors, the production process is not always center stage. Sometimes, people just want to admire the possibilities, or use them as a marketing stunt. However, some examples from other sectors would have never happened without a transparent and modular conception of the work: Wikipedia, *OhmyNews*, but also Zeroprestige – the “kite surfers” – and Lego would not exist without the possibility of working in a different way.

A thorough reform of software manufacturing, such as the community source propagated by IBM or the one practiced by Linus Torvalds for years, is necessary in order to fully convert to an open source mode of production. Modular system structures have been the trend in the software sector for some time now. This trend and the actual achievement of an open source culture are now merging; top-down management has served us well for years, but we have forgotten about bottom-up initiatives for far too long. To a large extent, this is a general management problem.

Most likely, publishing will follow the software sector in the conviction that better performances are possible with open source, so adopting it is therefore desirable. Just as in the software sector, the publishing sector is encountering stiff competition from sources outside the traditional in-

dustry: blogs and online newspapers are gaining in popularity. The software sector remains in the lead, but other sectors are following in its wake. How rapidly things will unravel strongly depends on the spread of expertise and skills outside the traditional core areas of the sector in question. The development of tools and simulation will also have a key role to play.

### Notes to Chapter 5

1. Tylor, Edward B., *Primitive Culture*. 2 vols. New York: Harper Torchbook, 1871.
2. [collaboration.csc.ncsu.edu/laurie/Papers/csed\\_AgilePractices.pdf](http://collaboration.csc.ncsu.edu/laurie/Papers/csed_AgilePractices.pdf).
3. Robles, G., J.J. Merelo & J.M. Gonzales-Barahona, *Self-organized Development in Libre Software Projects: A Model Based on the Stigmergy Concept*, 2005, [gsyc.escet.urjc.es/~grex/stigmergy-robles-merelo-barahona.pdf](http://gsyc.escet.urjc.es/~grex/stigmergy-robles-merelo-barahona.pdf).
4. Scacchi, W., "Understanding Requirements for Developing Open Source Software Design," *IEEE Proceedings – Software*, 2002, [www.ics.uci.edu/~wscacchi/Papers/New/Understanding-os-Requirements.pdf](http://www.ics.uci.edu/~wscacchi/Papers/New/Understanding-os-Requirements.pdf).
5. Loughborough University-led termites project, [www.sandkings.co.uk](http://www.sandkings.co.uk).
6. Raymond, E.S., *The Cathedral and the Bazaar*, version 3.0, 2000, [gnuwin.epfl.ch/articles/en/cathedralbazaar/cathedral-bazaar.pdf](http://gnuwin.epfl.ch/articles/en/cathedralbazaar/cathedral-bazaar.pdf).
7. [nl.wikipedia.org/wiki/Meritocratie](http://nl.wikipedia.org/wiki/Meritocratie).
8. Mockus, A., R.T. Fielding & J.T. Herbsleb, *Two Case Studies of Open Source Software Development: Apache and Mozilla*, 2002, [www.research.avayalabs.com/techreport/alr-2002-003-paper.pdf](http://www.research.avayalabs.com/techreport/alr-2002-003-paper.pdf).
9. Raymond, E.S., *The Magic Cauldron*, 1999, [www.catb.org/~esr/writings/magic-cauldron](http://www.catb.org/~esr/writings/magic-cauldron).
10. MacCormack A., J. Rusnak & C. Baldwin, *Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code*, Harvard Business School Working Paper Number 05-016, [open-source.mit.edu/papers/maccormackrusnakbaldwin2.pdf](http://open-source.mit.edu/papers/maccormackrusnakbaldwin2.pdf).
11. Worthington, D. "IBM Turns to Open Source Development," *BetaNews*, June 13, 2005, [www.betanews.com/article/print/%20ibm\\_Turns\\_to\\_Open\\_Source\\_Development/1118688437](http://www.betanews.com/article/print/%20ibm_Turns_to_Open_Source_Development/1118688437).
12. [www.sun.com/communitysource](http://www.sun.com/communitysource).
13. [www.cnn.com/tech/computing/9911/19/sun.comm.source.idg](http://www.cnn.com/tech/computing/9911/19/sun.comm.source.idg).
14. [alistair.cockburn.us/crystal/articles/cgm/cooperativegamemanifesto.html](http://alistair.cockburn.us/crystal/articles/cgm/cooperativegamemanifesto.html).
15. Cockburn, A. "The End of Software Engineering and the Start of Economic-Cooperative Gaming," *ComSIS* Vol 1, No 1, [www.comsis.fon.bg.ac.yu/Comsispdf/Volume01/InvitedPapers/AlistairCockburn.pdf](http://www.comsis.fon.bg.ac.yu/Comsispdf/Volume01/InvitedPapers/AlistairCockburn.pdf).
16. [www.itconversations.com/transcripts/175/transcript-print175-1.html](http://www.itconversations.com/transcripts/175/transcript-print175-1.html).
17. Cockburn, A., *Crystal Clear: A Human-Powered Methodology for Small Teams*, 2004.
18. Ibid. 17, zie ook [www.informit.com/articles/printerfriendly.asp?p=345009](http://www.informit.com/articles/printerfriendly.asp?p=345009).

19. Brooks, F.P.: “No Silver Bullet: Essence and Accidents of Software Engineering,” *IEEE Computer*, April 1987.
20. Royce, W.W., “Managing the Development of Large Software Systems,” *Proceedings IEEE WESCO*, [www.ctg.albany.edu/publications/reports/survey\\_of\\_sysdev/survey\\_of\\_sysdev.pdf](http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf).
21. Barnett, L. *Applying Open Source Processes in Corporate Development Organizations*, Forrester Research, 2004.

## 6 How open is the future?

In this chapter, we will sketch three scenarios of an open future. These three outline the major dilemma facing companies today: how do they protect their property rights while cashing in as much as possible on open innovation? At the end of this book, we will revisit the most relevant issues concerning open source and open innovation in order to raise the questions, “How open is the future? Is open source only a temporary phenomenon or are we standing on the eve of a paradigm shift in the way that organizations innovate and create value?”

*This chapter has been written with the assistance of Jason Haislmaier, a lecturer on copyright law at the University of Colorado and a partner in the Intellectual Property Group at the American law firm Holmes Roberts & Owen.*

Are we actually able to create a free and open culture where everyone forms communities for the purpose of participating in the creation of innovations? Are communities going to operate on their own in order to innovate and create increasing value in ways resembling those found in the software industry? Or are the legal restrictions so great that we will remain in the generally closed production culture that is now so familiar to us?

This is stating the issue in a rather black and white manner. In using the metaphors of open and closed, there is always the question of how open is “open” and does “closed” mean “shut up tight.” Surely there is a whole range of intermediate stages between the open and closed, states discernible in current open source practices? Ultimately, the dispute over ownership will be decisive in determining how open the future will be; this battle between public interest and private property is in full swing and its outcome remains uncertain.


### 6.1 Public versus private

We have grown up with the idea that companies make products and customers purchase them. However, in the open innovation model, customers are also the co-producers of these products. This gives rise to the question of who is then permitted to claim ownership of the creations: the companies or the customers. For open source software, this issue of ownership is resolved by means of new licenses ensuring that no one has ex-

clusive property rights to the products. They are in the public domain and, to some extent, actually belong to us all. However, the same licenses stipulate that ownership of anything contributed to the original product is relinquished by the contributor and automatically transfers to “us all.” Private property is consequently handed over to the community, and the public domain grows as a result. In this context, the public domain is deemed to comprise the knowledge and innovation (especially creative work such as music, inventions, art and books) in which no one can have a special interest. Everyone is entitled to use them, as they are presumed to be a part of our cultural heritage. What does not belong to the public domain is private property, and the use of intellectual property is therefore subject to restrictions.

In the face of what appears to be a communalization of ownership, many commercial enterprises have launched a counter offensive. They shore up more patents and copyright than ever. The legal opportunities to acquire such rights have been expanded in recent years and eager use has been made of them. While the opportunities for creating value with external parties and through open innovation are also increasing, many companies are falling back on rigorous protectionism of intellectual property. A collision between the protection of private property rights and the public interest appears to be inevitable. Therefore the question is whether or not the lawyers are going to win a victory over the freedom fighters.

The conflict does not concern a company’s right to a sizeable meal ticket. There are few people who question entitlement to profit. Even the most fanatical proponent of the open movement, the Free Software Foundation, believes that a company has a right to a return on its own R&D. The real conflict involves much more practical barriers. We are racing headlong down a path leading to a strict “permission culture” where we won’t be able to take a step without first checking whether we are infringing upon someone’s intellectual property rights and must request permission from all potential rights holders. Common sense appears to have vanished. The arrival of the internet and collaboration with “the community” now force us to review and perhaps amend the legal regime we’ve created. The conflict can have various outcomes: legal disputes, new legal constructions, collaboration, piracy, or a combination of all of these. Every day, we see that all four potential outcomes can occur side by side without any clear winner emerging.

The issue is essentially shaped by the permission culture in which we find ourselves. Lawrence Lessig, a professor at the Stanford Law School, has made it his life’s work to warn us against the advances of proprietary culture whereby the ownership claims of commercial companies are gaining a greater hold on our daily lives. Large, mostly American, lobby machines are perpetually busy staking ever larger claims on the basis of patents and copyrights that consequently extend the domains of private corporations. To escape this stranglehold, Lessig conceived of the so-called “creative commons,” marked by the  sign instead of ©. He created



an “open” license for books, images, ideas and creative work, modeled on the GPL license.

The answer to the question “How open is the future?” must, in the end, come from you. Openness is certainly the current trend. A milestone was reached in the case of *United States versus Causby* in which the American court dismissed the legal challenge of the Causby farm family. The Causbys contested the lawfulness of military aircraft flights over their land. Due to the noise, one hundred and fifty of their chickens flew against a wall and died, forcing the Causbys to give up their livelihood. The Causbys argued that, according to existing law,, the property of each parcel of land extends upward to the edge of the universe. The courts decided, however, that this view was out of date and that the public interest in aviation prevailed over their private interest.

“It is ancient doctrine that at common law ownership of the land extended to the periphery of the universe – *Cujus est solum ejus est usque ad coelum*. But that doctrine has no place in the modern world. The air is a public highway, as Congress has declared. Were that not true, every transcontinental flight would subject the operator to countless trespass suits. Common sense revolts at the idea. To recognize such private claims to the airspace would clog these highways, seriously interfere with their control and development in the public interest, and transfer into private ownership that to which only the public has a just claim.”

*United States vs. Causby*, 1946<sup>1</sup>

There are, says Lessig, comparable proprietary constructions that impede the free innovation of information and program codes. In the future, a court could again make a sudden correction to the balance between private property and public interest, only this time the decision might not benefit aviation but curtail all intellectual property rights, making it slightly easier for companies and individuals to engage in open innovation. Should this occur, it would provide an enormous impetus to innovation in general and our prosperity in particular. For Lawrence Lessig, such an end to the paralyzing effect of claims to intellectual property cannot come soon enough.

## 6.2 Neither mine nor yours, but public domain

In examining the open manner of doing business, our first task is, of course, to consider the discussions concerning the legal consequences of open source. After all, open source may be regarded as the focal point of open innovation, the model for other “open” developments. However, our primary concern in this chapter is to look beyond the boundaries of the open source software domain and pose questions about the ownership of

open innovation in general. We are not going to examine the various types of open source licenses in any detail. That information can be obtained from the internet site of the open source initiative.<sup>2</sup> Furthermore, the Wikipedia internet encyclopedia is another good source for a discussion of the ins and outs of open source licenses,<sup>3</sup> and a practical guide focused on risk management is provided by Jason Haislmaier's article "Closing the Open Source Compliance Gap."<sup>4</sup>

The essence of the discussion about open source licenses involves the question, "Who owns it, anyway?" This leads us to speak about a social development having consequences far beyond that of a license for a piece of software. It touches on the question of balancing interests in private property with what we can assign to the public domain. Society's interest in limitless enjoyment of software, music, written text and inventions comes into conflict, with increasing intensity, with corporate interests in earning income. The open source world and, in particular, the Free Software movement want to make the public domain as large as possible. The public interest in making free use of program code is forced on companies in the form of what we identify as a "viral" form of license. This means that private property added to something covered by such a license is transformed into public property. To explain this point further, let's reconsider the example of John Fluevog and his open source shoes, which we discussed in Chapter 2.

John Fluevog is one of the businessmen who have found a way to do business in a more open manner. Whoever visits the open source footwear page on his website will be invited to design shoes for his company. Anastasia in Russia was one of the people who responded, and an Anastasia shoe was, in fact, ultimately introduced to the market. But who owns this shoe model? Is Fluevog the owner because he has borne the production and marketing costs and added his own knowledge about making shoes? Is the owner Anastasia, who originally made the design and has rights to it on the basis of her intellectual property?

Who really cares about the intellectual property involving these shoes? Fluevog doesn't, as he has set the condition that every design will enter the public domain. Anyone making a submission must agree with the following statement:

"The submitting party understands and agrees that submission of this design to John Fluevog Shoes Ltd. releases the design into the public domain, where it is owned by no one and freely available to all. All claims to compensation in any form are waived."

John Fluevog

Consequently, Anastasia must renounce any copyright that she originally had due to the fact that she came up with the design. Were she the copy-

right owner, she would be able to demand that others wanting to make use of her intellectual property first obtain her permission. In relinquishing this right by releasing the design “into the public domain,” she ensures that no one has any exclusive right to its use. Not even John Fluevog, despite being the person inviting the submission of designs. Any other shoemakers can simply use the design to produce their own versions of the Anastasia shoes.

Fluevog isn’t worried about this possibility, probably because he doesn’t believe any other manufacturer would do that. However, there are other businesses that would be deeply concerned about such a possibility. Of course, we have to acknowledge that the legal construction of Fluevog’s open production method is far from watertight. The legal validity of simply stating that anyone submitting work automatically releases his or her intellectual property into the public domain is still unclear. In nearly all legal regimes, there continues to be an owner. Fluevog’s claim that the design is released into the public domain, “where it is owned by no one” is not legally binding and may not be enforceable. However, it is certainly possible for any owner (and, in this case Anastasia, the owner of the sketched shoe design) to relinquish her rights. Jason Haislmaier makes the following comparison:

“Imagine that you are the owner of a water fountain in the local park. You could decide to permit visitors access to the fountain when they want to have a picnic. At the same time, you could refuse to allow the construction workers working nearby to use the water in their cement mixers, or charge them a fee for doing so. Fundamentally, as the owner of the rights in the property, you can decide to grant or retain any ‘bundle’ of the rights to whomever you want. The same applies to the software licenses of proprietary manufacturers. Most of the rights are kept for the manufacturer itself and income is received for every bundle that is given away. Alternatively, the owner may retain only a few rights and make a broad contribution of rights to anyone in the software community for little or no fee. In certain cases, he may even do both simultaneously. This is how we end up with both proprietary and open source software, respectively.”

Jason Haislmaier

The specification “release to the community” is one we recognize from the “open source license” of open source software. In that case, the holder of the intellectual property specifies the rights of use. The open source initiative tests licenses on the basis of the principles of “openness.” All the licenses that can be used when one wants to “release” something to the community are listed on the website [www.opensource.org](http://www.opensource.org). We must keep in mind that under U.S. law the presumption is that computer software cannot be donated to the public domain, except in very narrow circumstances.<sup>5</sup>

There are several licenses grouped into two significantly distinct categories: the license roughly encapsulated by the statement “I’ve made this, do with it what you will” (under certain explicitly stated conditions), and licenses that are more fundamental and make it possible for anyone to add something to the intellectual property, an improvement that is then also released to the community. Let’s assume that Anastasia had issued her design under this freer type of license, of which the General Public License (GPL) is an example. Since a sketch is not a piece of software, the text in the license is therefore not fully equipped for these sorts of situations. However, there is also a GPL that is not intended for software but documents.<sup>6</sup> If this license were applied to the Anastasia design, it would mean that a conflict would immediately arise if the shoe sole was also patented. After all, the GPL says that everything you include in a product based on this license automatically comes under the license. This is what we call “viral”: something else is “infected” with the freedom that the license prescribes. Sections 7 and 8 of the GPL (version 2) indicate that, if there are other obligations – and a patent is such an obligation – the new product may not be sold.

Now suppose that Fluevog has a patent on a specially designed shoe sole, one that forms the basis of the entire product line. He then has a number of choices. He could relinquish the patent, like IBM and other companies whose patents were “released to the community.” He could sell the shoes in countries where the patent is not valid. Or he could abandon the entire plan.

In the case of software, IBM has donated five hundred patents to the open source community as part of a so-called “pledge.” Arrangements have been made so that neither IBM nor any other party can make any claim regarding the patents; they may be used free of any obligation.

#### **IBM’s 500 patent pledge<sup>7</sup>**

“IBM hereby commits not to assert any of the 500 U.S. patents listed below, as well as all counterparts of these patents issued in other countries, against the development, use or distribution of Open Source Software. In order to foster innovation and avoid the possibility that a party will take advantage of this pledge and then assert patents or other intellectual property rights of its own against Open Source Software, thereby limiting the freedom of IBM or any other Open Source Software developer to create innovative software programs, the commitment not to assert any of these 500 U.S. patents and all counterparts of these patents issued in other countries is irrevocable except that IBM reserves the right to terminate this patent pledge and commitment only with regard to any party who files a lawsuit asserting patents or other intellectual property rights against Open Source Software.”

Computer Associates has also made a similar pledge,<sup>8</sup> but when Sun Microsystems gave 1,600 patents to the open source community to mark the launch of the OpenSolaris operating system, the company kept the announcement to a simple statement.<sup>9</sup> This resulted in a great deal of criticism because, without an explicit pledge, there are a number of unanswered questions about the possibility that Sun may, in one way or another, make claims on its intellectual property at some time in the future.

Another risk, one that Fluevog is running when he does not protect himself legally, is the possibility that Anastasia may have stolen the design by simply copying it from a brochure or by setting it to paper after seeing it in a store. The true owner of the intellectual property could successfully take Fluevog to court. If the shoe is developed by one person, it is easy to protect oneself against this risk. Anastasia only needs to declare that she thought of the design herself. But if thousands of people have worked on a product and it is no longer possible to establish who has done what, things could become slightly more difficult.

For this reason, Richard Stallman asks all the people contributing to “his” GNU Linux to state that they were the original owners of their contributions and that they waive their ownership rights. Still, anyone working on the Linux kernel does not make any declaration that the code is not stolen. Therefore there is a chance that others might actually own parts of the code for a product thought to belong to the community. Fortunately, there are several companies who are willing to indemnify users against such claims: HP and IBM, to name two. Furthermore, it has also recently become possible to obtain insurance against such risk by means of the Open Source Risk Management program at Lloyd’s.

**First-ever Open Source Compliance Insurance now available through partnership between London-based Lloyd’s underwriter Kiln, Lloyd’s broker Miller and Open Source Risk Management**

London and New York – 31 October 2005 – Kiln PLC of London, U.K., a Lloyd’s of London underwriter and Miller Insurance Services Limited (Miller) a Lloyd’s broker, announced today that they will offer a new product called Open Source Compliance Insurance.

Press release from Open Source Risk Management, October 31, 2005

Jason Haislmaier emphasizes the fact that fear is a poor advisor and that full compliance with the conditions of open source can certainly be achieved. He is annoyed about the fact that much attention in many business seminars is devoted to the risks of open source. You only need to know what you are doing, be aware of the risks and act appropriately, just as in any other business activity. That is nothing new. Anyone wishing to pluck the fruit of open source in the technological environment and open

innovation in the business environment need only deal with the risks in a prudent manner.

The stories about the risks of open source can often be traced back to the most important legal action against open source: *Santa Cruz Operation (SCO) versus IBM*. A current account of this case can be found on [www.groklaw.net](http://www.groklaw.net). It is a dispute about the ownership of Linux, and it was summed up by Eben Moglen, professor at Columbia Law School and lawyer for the Free Software Foundation.

### **SCO versus IBM**

*Through a number of acquisitions, SCO came to own certain rights in the Unix operating system. When financial problems arose, new CEO Darl McBride decided to increase the revenue from its intellectual property.*

SCO made it clear to users that it had rights to a part of their software, and that it must be paid for user rights. IBM wanted to put a stop to such claims, which led to a lawsuit. As part of Linux, IBM released some of the source code that AT&T, the forerunner to SCO, had licensed to IBM for use in their AIX operating system. This is the bit of the code over which SCO claims ownership.

Moglen disputes the claim that IBM released SCO's property on two grounds. First, SCO has no rights to the code, despite having inherited it. Second, any claim would have to involve a carbon copy of the Unix code, which is not to be found in Linux. The codes of Unix and Linux bear a poor resemblance to each other.

Up to a certain point, it was just a dispute between SCO and IBM. However, SCO itself was also distributing the Linux program under a General Public License (GPL), which in fact says, "Do with it what you will." According to Moglen, SCO did not really expect to win the case, but was playing another game. SCO sent letters to all the largest companies saying, "If you are using the Linux kernel, you should be purchasing a license from us." That made SCO stock shoot up.

Unfortunately, SCO was also shooting itself in its own foot. By demanding money from companies for something that it had made but on which thousands of others had also worked, SCO was exposing itself to innumerable countersuits. In fact, IBM filed a counter claim against SCO due to its contravention of the GPL freedom clause. The only response that SCO could make to this was to deny the legal validity of the GPL.

Next, Microsoft gave SCO US\$10 million that, according to Moglen, was intended as an inducement to challenge the GPL. Consequently, SCO's McBride declared that the GPL would ruin the software sector, that it was un-American, and made similar polemical claims. However, in Moglen's opinion SCO does not have any possible grounds for proceeding against open source. In the most pessimistic scenario, one in which a bit of the Linux code is, in fact, found to be SCO property, the infringing elements could easily and quickly be replaced.

The SCO company has now been transformed from a software company into a company specializing in lawsuits. In December 2003, SCO shares were selling at nearly US\$20. Since Fall 2004, the share price has hovered around \$US4.<sup>10</sup>

SCO is trying to undermine the legal foundation of open source software production, while making a number of scaremongering statements about open source in general: it is anti-American, communist and something that hacks away at the roots of the capitalist system. Despite the hyperbole in such views, they clearly testify to the naivete of thinking that a closed world, in which the protection of intellectual property is an important basis for competitive strength, is simply going to shift to an “open” world, where everything belongs to everybody and everyone is free to build on existing products.

### 6.3 Permission culture: “open” under attack

If we can believe pessimists such as Lawrence Lessig, supporters of “open” are currently holding the losing hand. They are caught up in a battle among people in power, large multinationals and other companies that earn income from patents on discoveries and from other ways of securing intellectual property, such as copyright and trademarks.

The legal implications of “open” go right to the heart of “capitalism.” Industries, governments, multinationals, lawyers and militants for an open and free world are busy working on the question of how intellectual property can be legally put to bed. This involves a number of issues.

First, we must state that the internet has made it especially simple to distribute bits of digital information and, consequently, pieces of digital property in a quick and easy manner. Films, text, music and software programs move around the world without the owner earning a cent.

Second, it needs to be acknowledged that code is being generated in increasingly greater quantities. The tools enabling us to create digital products are more accessible: not only are they simpler to use and qualitatively better, but also less expensive. As a result, the opportunities of “open” production are also increasing; production is becoming distributed and is moving outside company walls. The intellectual property rights on creativity that has disparate origins is also becoming less tangible and so harder for companies to secure.

Finally, we must recognize that it is progressively easier to claim intellectual ownership in some regard. This creates barriers for other individuals and organizations, which must first make agreements with owners. For example, it has recently become possible to request a patent on human and animal genes. At present, 20 percent of genes have been patented. The improved efficiency in the United States Patent Office is also mentioned as another reason for the increase in the number of patents; it now has a stake in issuing as many patents as possible. Even universities have an increasingly sharp eye for the acquisition of patents. In the academic sector the pursuit of knowledge is increasingly a race for patent protection, despite the fact that free dissemination of knowledge remains



a basic principle of the scientific community. Lode Wyns, professor of biochemistry at the Free University of Berlin, also identifies this as a dangerous trend. In his view, it is absolutely absurd for universities to be concerned about the number of patents that they are able to acquire. In an interview included in the book *How Open is the Future?*<sup>11</sup> Wyns begins with a quote from Jonas Salk, who synthesized a polio vaccine in 1955. In response to the question whether he was going to patent his discovery, Salk said, “A patent? Could you patent the sun?” Salk regarded his discovery as something for the people, something that was in the public interest. Beginning in 1980, it became possible to acquire patents on living substances, and a veritable stampede resulted. In 2006, universities hold 3,600 patents. In 1965, the total was only 95 (see Figure 6.1).

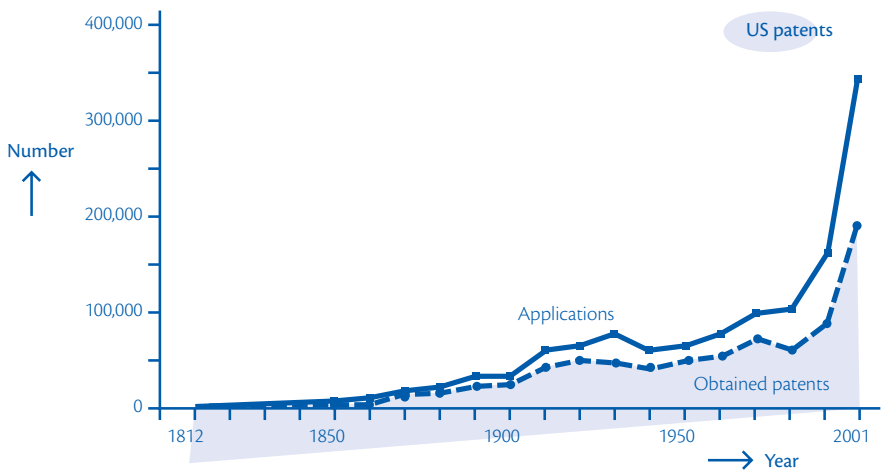


Figure 6.1: The registration of patents doubled during the internet decade

On the one hand we experience the emergence of an “open” movement, or an “open-source” movement if you wish. At the same time however the strategy of companies and universities in general remains predominantly “closed.” The remarkable growth in both the open and closed direction probably means that to date great progress is being made on various levels in many industrial and scientific areas.

The increased opportunities for open innovation have not led to a reduction in property protection. For the sake of convenience, suppose the number of patents is a key indicator for the closed model and the growth in the number of internet users an indicator of potential gains for the open model. In the last ten years of the twentieth century, we saw a doubling in the number of patents and, in a little more than ten years, the internet exceeding a billion users.

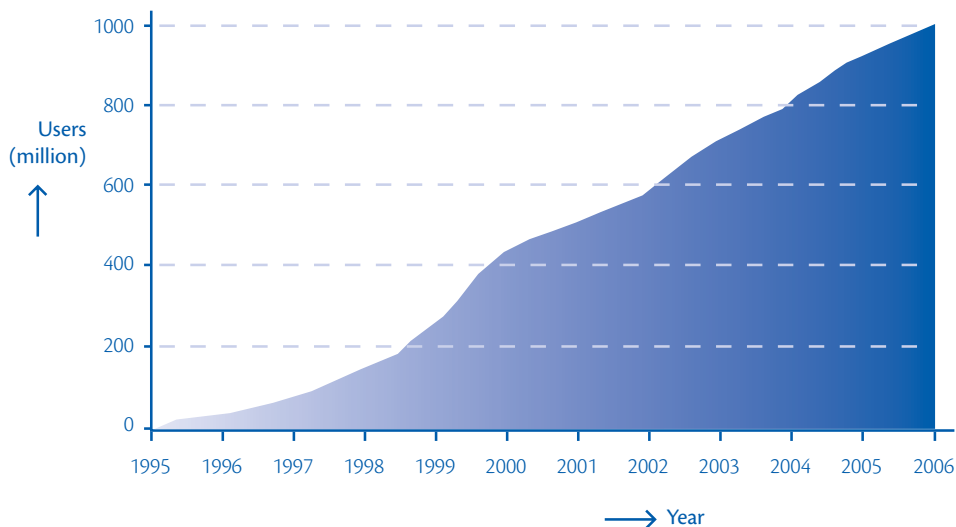


Figure 6.2: At the beginning of 2006, the number of internet users rose to 1.02 billion

In his book *Free Culture*, Lessig explains that corporate intellectual property is dominating our culture to an increasing degree. We find ourselves in a situation where we must ask permission every time we want to make something new. Lessig, who happens to lean in the direction of Richard Stallman's Free Software Foundation, believes that the entire situation has gotten out of hand, and intellectual property claims of companies have become entirely outlandish. With the arrival of the internet, the world has changed, and the new technology makes it necessary to amend the law. The overly pervasive culture of permission was created in America in 1976 when a new *Copyright Act* replaced the old one of 1909, and everything that was designed and produced automatically received a "copyright." Under previous versions of the *Copyright Act*, obtaining and maintaining copyright protection required the author or owner of the copyrighted work to observe certain formalities and requirements, including notice, publication and deposit. Under the 1976 *Copyright Act* and later amendments to it, many of these formalities and requirements were reduced or eliminated. As a result, copyright protection now attaches automatically to any work once the work has been created and fixed in a tangible medium of expression, and it is no longer necessary to apply for copyright – which up until that time was rare anyway.

We have seen that Lessig hopes for a legal decision, such as the one rendered in the 1946 case *United States versus Causby*. On that occasion, an American court ratified the act of Congress declaring the "territorial air column" above a piece of land to be a public right-of-way, despite its having been viewed differently in the past. Lessig refers to this case to argue that a decision to grant preference to the public domain over private

property is not anti-American or communist – quite the contrary. It promotes the economy and removes the barriers to healthy growth.

Turning to the internet, Lessig argues that the comparable situation existing there has not yet moved in the direction of common sense. We have arrived at an outrageous state of affairs in which we must ask permission for doing anything and using anything on the internet. We surf on the internet and create new things but, as if we were trespassing, must continuously ask permission from countless rights holders.

The patent race goes progressively further and, as mentioned, even as far as our genes. Lode Wyns uses research into breast cancer as an example of what can happen when private companies gain intellectual ownership of life itself. Once Myriad Genetics obtained a patent on genes associated with breast cancer, it decided to introduce the genetic test for breast cancer only to the American market. The price tag for the procedure was then set so high that not a single insurer agreed to pay for it. Still more serious, says Wyns, was the fact that the company does not conduct any research into a cure. Therefore the patent not only prevents the use of the knowledge in diagnosis, but also prevents the course of further research.

Wyns particularly objects to the fact that, although a great deal is said about these sorts of issues, very little is actually done. Intellectual property is acquired by the institutes with the most money. And the big money is in America. Wyns is concerned about the lack of support for more progressive ways of thinking:

“It looks as if the main forces are still heading the other way: against the current form of globalization, against free-market rules. Some sectors stick to the old rules and the open source spirit doesn’t have the means to enable them to achieve what they want. Our innovative potential is slipping towards third-world situations and we don’t know where it’s all going to end.”

Lode Wyns, 2005

Lode Wyns sees two ways of resolving this dilemma. The first is, persist. We cannot let the outrage over cases like Myriad Genetics to die down. The ongoing lawsuit might focus our indignation, but there’s no guarantee of a favorable outcome. There must be a general understanding that intellectual ownership cannot just be claimed by anyone. A culture in which knowledge is freely shared inside and outside of universities is productive. Wyns’s second solution is the catastrophe, as only a disaster will bring about real change. At present, things are so far off balance that such a disaster seems close at hand.

Lessig might seem to be an extreme pessimist, but his proposals for escaping the impasse are rather mild, especially if we compare them to

Wyns's catastrophe. Lessig's recommendations are specifically intended to take some of the bite out of permission culture, which would primarily entail reducing the tasks requiring lawyers and providing a few practical alternatives to the automatic "all rights reserved."

1. An alternative copyright. There are alternatives to the existing copyright. For software, there are open source licenses. For other content such as books, the "creative commons" exists as an alternative, recognizable by the double C within the circle instead of the single C of the traditional copyright. Lessig calls the creative commons a "reasonable" manner of establishing rights. No lawyers need to be involved when someone wants to do something with the content, and that is what Lessig sees as the greatest blessing: the reduction in transaction costs for those who want to work in this manner.
2. Shorter and simpler copyright. The duration of copyright should be as short as possible. Long enough to enable the maker to realize some benefit, but no longer. It must be simplified, for the interpretation of "fair use" or the distinction between "ideas" and "expressions" are things that keep lawyers busy.
3. The absence of a © symbol means "use unless...." As mentioned, copyright is automatically issued to the person who created the content. However, the absence of a copyright symbol on a work should mean that others need not ask permission to make use of the work.

## 6.4 Private plus public in the software sector

Chapter 3 described the development of the new mixed form of private and public property in the software industry. Self-interest and money have resulted in the convergence of private property and the public domain.

Such a mixture has proven to be possible in the software industry. Debian, a Linux variant, has a market value of US\$ 1.5 billion and is public domain. Companies like Red Hat earn substantial revenue from servicing open source products, products over which no one has exclusive control. However, there is a balance, a trade-off between public interest and private ownership. When organizations want to earn money by employing the public – people who are not on the payroll – the public asks for something in return, such as legal control, respect and even money.

When open source licenses are used, the collaboration between businesses and the community enlarges the public domain. However, many of these collaborations are not based on open source licenses. Ownership of what is then made and created is, in most cases, legally assigned to the companies. The makers receive a fee for their efforts or a share in the company's income if the product generates new business.

In “mash-ups,” both the public domain and private ownership grow, which is one of the reasons why these forms are strongly on the rise. A mash-up is a new service that consists of an existing webservice and something new that somebody has added to it. The APIs of Google Earth or Yahoo Maps are, accordingly, made available and anyone can build his or her own service around them. Other examples are provided in Chapter 2. Yahoo distributes its software development kit (SDK) under an open source license (BSD), but the services that you can build with it come under a special license (Terms of Service). Such service contracts assign most of the intellectual property to the company. Google does the same.

“As between You and Google, You acknowledge that Google owns all right, title and interest, including without limitation all Intellectual Property Rights, in and to the Service and that You shall not acquire any right, title, or interest in or to the Service, except as expressly set forth in the Terms of Use.”<sup>12</sup>

Bret Taylor, the brand manager of Google Local, says that the success of Google mash-ups is partly due to the fact that the public can make money with them. Google gives the mash-up makers a share of the revenue from advertisements that Google sells. As a certain mash-up proves to be popular, Google exercises the right to post advertisements on the site. (Over the third quarter of 2005, Google earned about US\$1.5 billion from online advertisement sales, a tripling of third-quarter earnings in 2004.) Consequently, rights and income are shared. Google has the right to place advertising; the makers have the right to use Google maps and to make something new from them.

Yahoo does indeed grant a portion of the intellectual property to the makers of new services on the basis of their APIs.

“Yahoo!’s rights apply to the Yahoo! APIs and all output and executables of the Yahoo! APIs, including Yahoo! search results, but excluding any software components developed by you which do not themselves incorporate the Yahoo! APIs or any output or executables of the Yahoo! APIs.”<sup>13</sup>

Instead of the frustration and lawsuits arising between owners of intellectual property and the “misusers” of it, a win-win situation is created. Makers of webservices, inventors who participate in the Innocentive network (see Chapter 2) were inspired to contribute something to someone else’s intellectual property and rewarded for doing so. The most important limitations of the permission culture as described by Lessig can be removed by clarifying in advance the agreement between the parties. Instead of regarding the sharing of self-built services as a threat, a new

business model is constructed and a new legal basis established to support a mutually profitable collaboration.

We are so concerned with construing the discussion about “open” in legal terms that we nearly forget to consider the other existing possibilities: ignoring the lawyers. For a lawyer, this is not an attractive option, but in practice, intellectual ownership is largely ignored in the case of books, music, games, computer programs and physical products. Immediately we think of countries like China, where protection of intellectual property is virtually non-existent. There is a joke going around about intellectual property rights in China: “China is the biggest fan of open source. Look at the number of Microsoft products being used there.”

In reality, things have gotten so out of hand that the statistics about the use of Linux in China cannot be right. PCs are sold with Linux, but as soon as a box is opened and the computer turned on, an illegal version of Windows is installed.

The defenders of private property, the large multinationals, are seeking a way to collar lawbreakers. One of their efforts involves closing down networks such as Napster on which digital files can be shared. The problem with this approach (or perhaps “tragedy” is the better word) is that the instrument is often very blunt. It not only prevents the illegal use of such programs, but often permissible practices are unnecessarily restrained. This draws the ire of Lawrence Lessig, who claims that the “all rights reserved” lobby now has the upper hand.

## 6.5 Conclusion

Neither open nor closed but “something in-between” seems the most likely scenario for innovation. We can see this happening in the software sector. However, it must also be said that open innovation and the public domain have much greater significance there than in other sectors. If the happenings in the software sector are to be mirrored elsewhere, the public domain will have to gain in strength in order to provide a better balance. The growth of the public domain in the software sector has occurred in a bottom-up manner: originating with users rather than with companies, and the GNU General Public License was subsequently the condition for this growth of the public domain. Correspondingly, we note that translations of property rights in the form of the “creative commons” are also percolating through to other sectors.

If the public domain is to prevail over private property, or at least to grow in size, the government will have to create the appropriate conditions. Despite all the open source initiatives, it is clear that other sectors are still unable to enjoy the same latitude that holds sway in the software industry. We will therefore conclude this book with a subject that we have not yet addressed: what role should government play? For the answer to

this question we can return to the farming couple who wanted to protect their property, their airspace. The court amended the law underlying their argument because a new technology required it. Similarly, outdated legislation is obstructing the production revolution sparked by the internet, the possibility of linking together all the intelligence all over the world. New production companies require protection to grow. When protectionism occurred in the aviation industry, the public domain increased in size. A flourishing new internet industry has also been aided by a larger public domain. As protector of the public sphere, the government has to create the appropriate conditions by re-evaluating existing legislation on intellectual property in the light of new technological possibilities, private intellectual property, and public interest.

However, this is not to conclude that we ought to wait for what the government will do before reaping the rewards of open innovation. Every company can begin experimenting with open innovation even now. The more success that companies enjoy with open innovation, the greater the chance that the future will, in fact, be open.

#### Notes to Chapter 6

1. [home.netvista.net/~hpb/cases/causby-1.html](http://home.netvista.net/~hpb/cases/causby-1.html).
2. [www.opensource.org](http://www.opensource.org).
3. [en.wikipedia.org/wiki/Open-source\\_license](http://en.wikipedia.org/wiki/Open-source_license).
4. Haislmaier, J.D. *Closing the Open Source Compliance Gap*, 2005, [www.hro.com/pubs/closingthegap.pdf](http://www.hro.com/pubs/closingthegap.pdf).
5. 37 cfr 201.36, zie [www.sice.oas.org/int\\_prop/nat\\_leg/Usa/rcref.asp#201.26](http://www.sice.oas.org/int_prop/nat_leg/Usa/rcref.asp#201.26).
6. [www.fsf.org/licensing/licenses/index\\_html#DocumentationLicenses](http://www.fsf.org/licensing/licenses/index_html#DocumentationLicenses).
7. IBM Statement of Non-Assertion of Named Patents Against OSS, [www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf#search='ibm%20Patent%20pledge'](http://www.ibm.com/ibm/licensing/patents/pledgedpatents.pdf#search='ibm%20Patent%20pledge').
8. Computer Associates International, Inc.'s Statement of Non-Assertion of Named Patents Against OSS, [ca.com/patents/oss](http://ca.com/patents/oss).
9. Sun Microsystems, *Sun Grants Global Open Source Community Access to More than 1,600 Patents*, January 25, 2005, [www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.2.html](http://www.sun.com/smi/Press/sunflash/2005-01/sunflash.20050125.2.html).
10. 'SCO vs. IBM', December 2003, [www.itconversations.com/shows/detail61.html](http://www.itconversations.com/shows/detail61.html). (The entire analysis is available for listening or to be downloaded as a podcast)
11. Wynants, M., Cornelis, J. (Eds.) *How Open is the Future? Economic, Social & Cultural Scenarios Inspired by Free & Open-Source Software*, VUB Brussels University Press, 2005.
12. Google Maps API Terms of Use, [www.google.com/apis/maps/terms.html](http://www.google.com/apis/maps/terms.html).
13. *Yahoo SDK Software License Agreements and Web API's Terms of Service*, [developer.yahoo.net/download](http://developer.yahoo.net/download).



## 7 Thirteen lessons for open innovation

“The aim and the core of this book is to draw inspiration from developments in open source software and apply it profitably to business innovation.” This is how the first chapter begins. We hope that the spread of open source outside the software sector and the stormy impact of open source culture inside of it have provided you with sufficient incentive to experiment further with open innovation.

The extremely important connection between open innovation in all sectors of the economy and open source is being made to an increasing extent, as illustrated by the Demos report “Wide Open: Open Source Methods and Their Future Potential” and by the vision of disruptive innovation propagated by Professor Steven Weber. That is why a passage from the Demos report and an expression of Weber’s enthusiastic expectations appear as epigraphs for this book. The connection is also made by the reputable consulting firm McKinsey & Company, which issued a statement in the spring of 2006:

“Thanks to the many books on ‘open innovation’ and to the prominence of open-source software projects such as Linux, most executives have at least a passing familiarity with the subject. Its central idea is that when companies look outside their own boundaries, they can gain better access to ideas, knowledge, and technology than they would have if they relied solely on their own resources. Some executives may even be familiar with the many variants of open innovation, a number of which stray a considerable distance from traditional ‘closed’ models of innovation management. Despite the familiarity of these ideas, persistent doubts and misunderstandings often make it hard to generate value from them. At one extreme, many people ask whether distributed models of innovation aren’t notoriously hard to control, manage, and commercialize. At the other extreme, open innovation may seem to be mostly about narrowly defined joint ventures or transactions to acquire intellectual property created by others. If so, what’s all the fuss about? In truth, except for narrowly scoped forays (such as the licensing of technology) outside the confines of the enterprise, few top executives believe that they understand how best to create value with the open model of innovation.”

McKinsey, 2006<sup>1</sup>

Despite the success of user-led open source methods in the software industry, there is only wavering belief that open innovation can create value for individual businesses in other sectors. To build a more obvious bridge between open source software and other fields where open innovation could be applied, we have provided numerous examples of open innovation, intended as sources of inspiration.

These examples show that the inescapable future of open source and the inevitable emergence of a new production model (open innovation) will converge and reinforce each other. This insight should provide an important stimulus, making it clear that business can no longer wait to react to developments in their own markets. Making a proactive early start down the path of open innovation will increase the chance of acquiring a competitive advantage.

We will therefore conclude this book with thirteen lessons for open innovation, in order to help you sharpen your strategy along the suggested lines.

#### **1 The end of the closed business model**

Open innovation provides a rich array of possibilities. Companies that close themselves off to these opportunities will, to an increasing extent, be confronted by organizations that are, in fact, profiting from open innovation. Completely closed companies will lose ground to the competition and they will find it increasingly difficult to keep up. It is no longer a question of IF but WHEN to adopt open innovation.

#### **2 Future competition will come from anyone with an idea**

Opportunistic competitors will primarily succeed by binding people to the organization. In a time of hypercompetition, there is less and less time to nurture good ideas and allow them to develop. Before we know it, the competition has outsmarted us. At Procter & Gamble, the definition of competition has been updated to be more in tune with modern times: “My biggest competitor today is a person with an idea.” Switching from consumer products to software production, we see the benefits of user involvement described by Eric Raymond in *The Cathedral and the Bazaar*: “Given enough eyeballs, all bugs are shallow.” Anyone can come up with ideas for improvement, and a software community makes it possible to gain access to the ideas of everyone. At Boeing, the community devoted to ideas for improvement is called the World Design Team. Anyone reading through the discussions published on the Boeing site will find such remarks as, “Have you thought about this possibility” or “I’ve noticed that you have thought about this, but would it not be smarter to approach the problem in such and such a manner?” This is not so different from the detection of software bugs, except that product improvements lead in this case to a better aircraft.

### **3 User experience is essential to open innovation**

If open source makes us aware of anything, it's the fact that users (customers) are essential and their experiences are priceless. Look at what happened with the experiences of a few kite surfers who decided to build their own kites. They were able to make a better product than those being sold in the store. Clearly, open innovation not only involves the quantity of ideas that companies can get their hands on but also the quality of these ideas. Buyers and users of products are better able than anyone to make evaluations based on experiences – better able even than the producer, who is no expert in that field.

For quite some time, many consumers have been wanting to modify standard products. A study on this subject, tracking consumer behavior over more than thirty years, reveals that on average 10 to nearly 40 percent of users have customized something in some way. Open innovation is now getting so much attention because the possibilities of doing something with the innovation process have now quickly and radically changed. The internet is the most important factor in this regard; open source would have a much more marginal existence if the internet did not exist.

### **4 Technology makes open innovation possible**

Thanks to the internet, we now have open source, weblogs, peer-to-peer communication and open innovation. All these forms of communication have changed society. We can do more and know more collectively than we can as individuals. This emancipates us, as knowledge and experiences can literally shoot off in all directions and are more accessible than ever.

One other aspect of technology is just as important for the rapid and radical emergence of open innovation: the tools that make it possible to co-create. An internet site for the development of software is one such tool, but so is a program for processing photographs. Everything that makes it possible for us to code and process the world around us contributes to open innovation.

These developments come together in a generation that is familiar with the internet, adept at using the tools and capable of transforming society by sharing its creativity with everyone. This generation is known as Generation C; the “C” standing for content, creativity, community and, of course, coding.

### **5 Take a digital view of the organization**

The languages of open innovation are digital. Discover the possibilities by adopting a digital view of your organization. Webservices, software, text, audio, CAD/CAM designs, but also just the simple communication tools of the open source community are all steps toward open innovation.

## **6 Find people with a passion for the product**

Open source software development has demonstrated that passion is an important factor for success. We see the same thing elsewhere: beer lovers got Brewtopia off the ground, aviation enthusiasts have – in a certain sense – supported Boeing and, who knows, there might soon be a new line of fishing rods on the market designed by passionate fishing enthusiasts. The true passion for a product can lead to something successful, such as Linux. Yet, it can begin with something as small as a single shoe, like the shoe developed by Anastasia in Russia for a company wanting to bring open source shoes to the market. Many creative individuals have a passion and are prepared to share it with each other or with a company. The path leading to profiting from such a resource involves binding the most important users – the “lead users” – more strongly to the organization. This can be a step toward a more open strategy for a company and the founding of a truly passionate community.

## **7 Anyone wanting to undertake open innovation must surrender partial control**

People from outside the organization are less willing to be managed. This is true in open source software projects and applies to other open innovation initiatives as well. To put it even more strongly, a part of the success depends on the extent to which people can manage themselves. Task allocation is at odds with this feature. Self-determination not only ensures that an efficient mechanism regulates the supply and demand of work, but it even leads to the best possible performances by participants, who are exceptionally motivated. In addition, aspiring participants estimate to the best of their abilities whether and how they can make a suitable contribution.

## **8 Open innovation makes fewer managers necessary**

Karim Lakhani is responsible for the MIT Open Source Research Community and he is writing a dissertation in which he speculates about what added value managers now provide. Open source production can be performed with very little overhead and the persons who assume some of the roles of bosses do so purely on the basis of their merits. They all are perfectly familiar with what has to be achieved because they have themselves come up through the ranks, and if they do not perform well, there is always someone else ready to take over. This is perhaps an idealized picture, but every manager should nevertheless understand that you can succeed in delivering very good performances in this manner. Why do we actually need managers is one of the many questions that open source raises. It may be that Lakhani’s question will inspire organizations to undertake more open innovation. The success of self-determination in open source software could cause a resurgence of self-directed teams and re-generate principles of self-direction that originally began to emerge in the eighties.

Outside the software sector, companies like Toyota have very seriously incorporated self-management principles into their own organizations

## **9 Open innovation is good for the company image**

Both inside and outside the software sector, we see that open innovation involves more than the initial question in this book: “How do we gain access to the minds of all clever people?” After all, having users and consumers collaborate with companies blurs the boundary between R&D and commerce. This was abundantly clear with Brewtopia, Threadless and Lego, as well as with the mash-up webservices of Amazon, eBay, Google and Yahoo, and also in the case of Sun’s Solaris operating system.

One of the attractions of open source is that it has a positive effect on company image. Open source is more readily associated with “good” whereas closed is associated with “bad.” IBM makes commercials for Linux, a product that does not at all belong to IBM, purely for marketing purposes. IBM will benefit if Linux becomes a strong brand, but it will also benefit by attaching itself to the successful image of this brand name. The same applies to “popular brands.” Google, eBay, Amazon and Yahoo all have a cast-iron image thanks to the latitude with which they invite customers to participate. This does not always involve open source in the strictest sense, but certainly open innovation. In particular, this involves brand democratization, the establishment of an open brand, which is not exactly the same thing as production democratization, the powerful effect of open source insofar as product innovation is concerned. Fluevog, Boeing, Converse and all the other examples of open innovation discussed earlier are certainly also concerned with “brand democracy.”

## **10 Slightly open is also permissible**

Closed and open are not mutually exclusive. The choice is not black or white. *Open for Business* can be realized in many ways. An analysis of what IT corporations do with open source reveals that “open” and “commercial” can reinforce each other. The pure open source players, such as Apache, are now associated with commercial open source enterprises. A hybrid software market has been created in which open and closed supplement each other. Open source has become mainstream, and every company wants to get a piece of the pie. The same thing is happening outside the traditional IT sector. In the most closely related field, new-economy companies like Amazon have discovered that sharing software with the external world can reap rewards. Estimates are that ten million requests are made to Amazon daily by way of webservices on other sites. This generates more traffic at the company’s internet store. An open innovation approach has proven successful, and demonstrates that a mixture of open and closed can work. *Open for Business* comprises several options: wide open, half open or open just a bit. It all depends on the desired relationship between open and closed.

**11 Intellectual property is to be protected, even in the case of open innovation**

Making money from R&D is still possible if rights are properly managed. See for example the licenses that Google uses for its webservices or the contracts used on the Innocentive innovation network.

**12 The open future offers a range of opportunities**

The developments in the software market indicate that open source is gaining the upper hand. At the same time, we see that legislation continues to place restrictions on “open.” IBM releases patents while simultaneously acquiring new ones. People like Lawrence Lessig and Lode Wyls prophesy a troubled future. The most important limitation on economic growth is the permission culture in which we are all now living. The domain of private intellectual property is growing larger, and the free exchange of knowledge in the public domain is being increasingly frustrated. Lessig and Wyls hope that common sense will prevail, but only time will tell when and how this might happen.

**13 Open innovation is just that good**

It is no accident that open source developers are able to make outstanding products. Specific attributes of open innovation are responsible for generating exceptional motivation and supreme focus. Open innovation allows one to enter the state of “flow,” the condition under which performance is optimized.

**Note to Chapter 7**

1. “Creation Nets: Getting the Most from Open Innovation,” *The McKinsey Quarterly*, April 2006, [www.cfo.com/article.cfm/6937262?f=related](http://www.cfo.com/article.cfm/6937262?f=related).

# Index

.NET 129  
3M 31, 46, 86

## A

Aalbers, Martine 142  
Accenture 91  
Adobe 104  
Adriaans, Pieter 26  
advertising 74-75, 89  
Agile Manifesto 162  
agile programming 152, 162  
agile software development 158, 161-166, 168  
aircraft construction 53-54  
Aisin Seika 68  
Allchin, Jim 160  
Amazon 42, 62-65, 86-87, 90, 193  
    Mechanical Turk 30, 66  
Amdahl, Gene 164  
AMOS 50  
Apache 15, 17-18, 21, 33, 37, 101, 114, 129, 145, 157, 193  
    Open for Business Project 28, 31  
    usenet 139-140  
Apple TV 111  
Artificial Intelligence Laboratory (MIT) 18  
associates 65, 90  
Asterisk 103  
Atos Origin 91  
Audi 75, 86-87, 89  
Augustin, Larry 100, 100, 122

## B

Balter, Dave 77  
Basic Linear Algebra Subroutines 131-132  
BEA systems 112  
beer 77  
Behlendorf, Brian 17-19, 101



Berkeley 17  
bio-computing 42, 50  
Birney, Ewan 50-51  
Bitzer, Jürgen 141  
Blaauw, Gerrit 164  
Blender 142  
blogging 30  
Boeing 80, 91, 123, 190, 192-193  
Boodman, Aaron 63  
Bordieu, Pierre 139  
Boston Consulting Group 68, 70  
brand democracy 193  
Brauer, Klaus 80  
Brewtopia 77-79, 87, 89, 192-193  
Brooks, Frederick 164-165  
Building an Open Source Space Program 54  
business 16  
business model, closed ~ 190  
Business Process Re-engineering 70  
buzz agent 77  
BzzAgent 42, 76-77

## C

CAD/CAM 42-43, 48, 50, 54, 57  
Cambrian House 30, 82-83, 86  
Canon 60  
Capgemini 91  
*Cathedral and the Bazaar, The* 154, 190  
*Causby, United States versus ~* 175, 183  
CDDL 119-120  
Center for Open Innovation 23  
Centraal Beheer (insurance company) 76  
challenges 129  
Cherkoff, James 75  
Chesbrough, Henry 23, 26, 70  
Chicagocrime.org 64, 86-87  
China 187  
Christensen, Clayton 34, 41, 44-45, 87  
Chumby 32, 83, 86  
Clickworks (NASA) 66-67  
Cloudscape 114  
Cockburn, Alistair 152, 161-164, 169  
Cohen, Josh 107  
CollabNet 18-19, 101, 132  
Colony, George 106

- commercial success 19
- commercials 75-76
- Committee for Economic Development 23
- Common Development and Distribution License (CDDL) 119
- common source 22
- communal character 14
- community 14, 36
- community product 89
- community source 22, 151, 158-159
- community-building process 154
- competition 124, 145, 190
- Compiere 16, 91, 97, 105, 113
- computers, building ~ 80
- computing 52
- Converse 75, 86-87, 193
- cooperative gaming 169
- copyleft 17, 27
- copyright 17-18, 174, 176, 185
- Copyright Act* 183
- cost reductions 122
- creative commons 49, 77, 174, 185, 187
- crowdsourcing 13, 27-30, 33
- CrowdSpirit 82-83, 86
- Crystal Clear 163
- Csikszentmihalyi, Mihaly 142-143, 146
- culture 151, 153
  - ~ of experience 158

## D

- David, Lionel 82
- Debian 100, 116, 185
- Dell 91, 106
- Democratizing Innovation* 46, 48, 77, 89, 151
- Demos 47, 189
- Digital Connections Council (DCC) 23-24
- Digital Designer (Lego) 55
- digital view of organization 191
- Digium 103
- Directorate-General for the Information Community (EU) 134
- disruption:
  - low-end ~ 45
  - new-market ~ 45
- Donofrio, Nicholas 47
- DoubleClick 111

Draper Fisher Jurvetson 104  
Dynamics 3.0 111

## E

eBay 63, 65, 193  
Eclipse 114, 129  
Eli Lilly 71-72  
Ellison, Larry 111  
Encyclopedia Britannica 84-85  
Epiphany 104  
ERP software 16  
Evans, Philip 68-69  
Evans Data Corporation 132  
experience, culture of ~ 158  
*Experience Economy, The* 80  
extreme programming 152

## F

F/OSS 99  
Factory (Lego) 55-56  
factory simulation 52-53  
family affair 140-141  
Fink, Martin 119-120  
Firefox 129  
FL/OSS 99  
Flexsim 53  
Florida, Richard 60  
flow 127, 142-146, 148  
*Flow: The Psychology of the Optimal Experience* 142  
Fluevog, John 58-60, 87, 122, 176-179, 193  
Ford, Henry 17, 43  
Forrester Research 106, 166  
*Free Culture* 183  
free software 21, 99  
Free Software Foundation (FSF) 17-19, 100, 174  
*Future of Competition, The* 46  
future scenarios 173

## G

Gabriel, Richard 30, 118  
Gates, Bill 37, 111, 128, 134  
General Motors 76  
General Public License (GPL) 17, 120, 178, 187  
Generation C 35, 41-42, 57-58, 60, 80, 191

- genetics 42, 50
- Gerlach, Charles 54-55, 80
- Gerlach Space Systems 54
- Gerstner, Lou 114
- Gerwen, Frans van 76
- gift economy 128, 138-141, 148
- Gigaworld 106
- Gilmore, James 80
- Global Innovation Jam (IBM) 31
- Globus project 114
- Gluecode 114
- Gnome 103
- Goldman Sachs School of Public Policy (University of California) 51
- Goldman, Ron 30, 118
- Gonzales-Barahona, Jesus M. 154
- Google 62-65, 90, 110-111, 185, 193-194
  - ~ effect 62
- Google Earth 185
- Google Local 185
- Google Maps 64
- GotDotNet 107-108
- Goto, Kazushige 131-132
- GPL 17, 19, 21-22, 81, 120, 175, 178
- Grassé, Pierre-Paul 154
- Greasemonkey 63-64
- Grosfeld, Thomas 24

## H

- Haislmaier, Jason 173, 176-177, 179
- Halloween Document 106-107, 109
- Hansen, Mark 55
- hardware 81
  - proprietary ~ 106
- hardware virtualization 52-53
- Harvard Business School 34, 44
- Hawk, Jeff 80
- Heintzmann, Doug 159-160
- Hippel, Eric von 19, 25-26, 34, 46, 48-49, 77, 88-89, 139-140, 151
- Horn, Paul 24
- Hot Wired* 17
- How Open is the Future?* 182
- Howe, Jeff 29
- HP 22, 91, 99, 106, 116-117, 120, 124
- human network, ingredients for succes 70

Hunter, Jason 131, 145  
hybrid model 22  
hype cycle (Gartner) 33

## I

IBM 15-16, 22, 35-37, 91, 99, 106, 112, 114-115, 120, 122-124, 151, 153, 159-161, 178, 193  
    Global Innovation Jam 31  
    SCO versus ~ 180-181  
IDC 62-63  
image 123, 193  
information manipulation 52  
Innobase 112, 125  
Innocentive 71-72, 86, 185, 194  
innovation 16-17, 22, 43  
    closed ~ 24-25  
    consumer-led ~ 25, 35, 46  
    disruptive ~ 34, 44-45, 86  
    open ~ 13, 23-25, 25-26, 33, 41, 189  
    user-driven ~ 27-28, 34, 46, 48  
    user-led ~ 151  
Innovation and Entrepreneurship Group (MIT Sloan School of Management) 34, 48  
innovation community 70  
*Innovation Happens Elsewhere* 30, 118  
innovation niche 49  
Innovation Platform (Dutch) 24  
*Innovator's Dilemma, The* 45  
Institute for Business Value (IBM) 54  
Institute for Software Research (University of California) 37, 152  
International Society for Computational Biology 50  
internet 17, 19  
    number of users 182  
Iridium 53

## J

Jager, Durk 71  
Janke, Jorg 113  
Java 121, 123  
JBoss 15, 19, 106, 112  
Jones Soda 77, 89  
Joy, Bill 159

## K

Kanenko, Steve 80  
Kelly, John 35, 115  
Kempelen, Wolfgang von 66

kite surfing 48-49, 54  
knowledge, internal and external ~ 123  
KRO (broadcaster) 58  
Krogh, Georg von 140

## L

Lafley, Alan 71  
Lakhani, Karim 19, 139-140, 142, 145, 158, 192  
language 88-89  
Lead User Concepts 46, 48, 151  
lead users 47, 86, 192  
Leadusers (website) 46  
learning process 144  
Lego 55-56, 86-87, 89, 91, 122, 193  
    open innovation lessons 57  
Lessig, Lawrence 115, 174-175, 181, 183-185, 187, 194  
Levine, Paul 65  
license 17-18, 22, 49, 77, 81, 119, 124, 174-177  
    categories 178  
Linux 15-17, 19, 21-22, 33, 35-37, 96-97, 99-100, 102, 105, 108, 114, 116-117, 119-120,  
    157, 187, 192-193  
    ownership 179  
Longhorn 37, 159-161  
Lotus Notes 159-160  
low-end disruption 45  
Lulu.com 58

## M

managers 192  
Mandrake 116  
MapPoint 111  
market innovation 17  
marketing 74-75, 89  
mash-up 42, 63-65, 111, 185, 193  
Maurer, Stephen 51  
McKinsey 189  
McManus, Jeffrey 63  
Mechanical Turk (Amazon) 30, 66  
Medellin, Rico 143  
medicine 42  
Merejo, Juan Julian 154  
meritocracy 157  
meritocratic leadership 21  
Mickos, Marten 101

Microsoft 16, 22, 35-37, 80, 91, 97, 105-111, 151, 153, 160-161  
    competitors 108-111  
    Google as enemy 110-111  
modular structure 151, 158, 170  
Moglen, Eben 180  
money, making ~ 96, 134-135, 148  
MONO 129  
motivation 127, 146-148  
    intrinsic ~ 19, 146-147  
Mozilla 99, 104, 129, 159  
MTV 58  
Mulhall, Liam 78  
Muller, Werner 71  
Myriad Genetics 184  
MySQL 16, 19, 36, 97, 99, 101, 105, 112-113

## N

Napster 187  
NASA 16, 30, 55  
    Clickworks 66-67  
National Center of Supercomputing Applications (NCSA) 17  
Netscape 99  
new-market disruption 45  
Nietzsche, Friedrich 136  
Nike 75, 89  
NineSigma 71-72, 91  
Nokia 26  
Novell 91, 99-100, 102-103, 112, 120

## O

Oh Yeon Ho 73-74  
*OhmyNews* 73-74, 85-87, 89  
Open for Business 28  
Open for Business Project (Apache) 28, 31  
open innovation 13, 23-25, 25-26, 33, 41  
    examples 42  
    lessons 189  
Open Innovation 70  
open software development 168  
open source:  
    characteristics 18  
    many faces of ~ 15  
    motivation 127, 146-147  
    power of ~ 16



- relationship with open innovation 13
- strategies 95-96
- Open Source Business Conference 44, 81
- open source community:
  - motivation 127, 147-148
  - roles 131, 153
- open source culture 151, 153
- open source developers:
  - characteristics 132-133
  - motivation 127, 147-148
  - wanting to belong 136-138
- Open Source Development Lab (OSDL) 100
- open source effect 62
- Open Source Footware 59
- Open Source Initiative (OSI) 19, 21, 119
- Open Source Reference architecture (HP) 116-117
- Open Source Research Community (MIT) 192
- Open Source Risk Management 96, 179
- Open Source Software Developer Lab 105
- Open Source Space Program 80
- OpenSolaris 118-119, 179
- OpenView (HP) 116
- Oracle 22, 91, 105, 111-113, 125
- Oran, Clint 104
- O'Reilly, Tim 45, 63
- Ormala, Erkki 26
- Overton prize 50-51
- ownership 173

## P

- Palmisano, Sam 115, 120
- Paraz, Migs 130-131
- passion for product 192
- patent pledge 178-179
- patents 14, 26, 174, 178, 181, 184
  - number of ~ 182
- Pathfinder 16, 55
- patron 124
- peer-to-peer 31-32
- Perl 129
- permission culture 174, 181, 183
- Permissive License 111
- pharmaceutical research 14, 50
- Philips 24-26, 31, 46

PHP 102, 112, 129  
Pichler, Roman 164  
Pine, Joseph 80  
Planeshift 157  
Plato 48  
PostgreSQL 105  
Powell, Julie 58  
Prahalad, C.K. 46  
press 73  
*Primitive Culture* 153  
private plus public 185, 185, 187  
Procter & Gamble 14, 42, 70-72, 190  
product innovation 50  
property:  
    intellectual ~ 26, 70, 174, 183, 185, 194  
    private plus public 185, 185, 187  
    public versus private 173-175  
proprietary hardware 106  
proprietary services 106  
proprietary-software suppliers 105  
public:  
    ~ plus private 175  
    ~ versus private 173-175  
public domain 175-177, 187

## R

R&D 13, 24, 122  
RAD 160  
Ramaswamy, Venkatram 46  
Rapid Application Development (RAD) 160  
Raymond, Eric 18-19, 154, 156-158, 190  
Red Hat 35-37, 91, 96, 100, 102, 112, 116, 120  
*Republic, The* 48  
Rhino 50  
*Rise of the Creative Class, The* 60  
Roberts, John 104  
Robles, Gregorio 154

## S

Salesforce.com 104  
Salk, Jonas 182  
*Santa Cruz Operation (SCO) versus IBM* 180-181  
SAP 22, 91, 102, 105, 112  
Scacchi, Walt 37, 152, 154, 156-157, 170  
Schrettl, Wolfram 141

Schumpeter, Joseph 34-35, 43-45  
Schwartz, Jonathan 81, 115, 120-121  
Science and Technology Policy Council AWT (Dutch) 25-26  
SCO versus IBM 180-181  
Second Life 31-32  
seeker-solver network 71  
seeker 91  
self-determination 19, 192  
self-direction 157  
self-management 147-148, 156  
Serviceguard (HP) 116  
shared source 22  
Shared Source (Microsoft) 107, 111, 151  
shoes 59, 176, 192  
Siebel 100, 104  
Siemens 164  
Simics 53  
simulation 52-53  
Simulation Based Design 54  
Sleepycat 112  
software development 166-170  
    role users 17  
software development sector 158  
software engineering 161, 164  
Software Engineering Institute 152, 158  
software market 95  
software portfolio 97  
Solaris 22, 97, 106, 115, 118-120, 123-124, 193  
solver 91  
source code 17-18  
SourceForge 19, 50, 108, 132, 141  
Spaeth, Sebastian 140  
Spencer, Mark 103  
Srivastava, Amitabh 161  
Stallman, Richard 17-18, 22, 100, 179  
Stanford University 132-133, 135, 139  
*Starzine* 58  
stigmergy 154, 156  
Stolena3.com 75  
*Success of Open Source, The* 17  
SugarCRM 36, 91, 97, 104  
Sun 22, 36, 81, 91, 97, 99, 106, 112, 115, 117-121, 123-124, 151, 159, 179, 193  
Sun Grid 121  
surfkite 86, 90  
Suse 91, 100, 102, 116

## T

Taylor, Brad 185  
Taylor, Bret 63  
Taylor, Jacob 104  
Taylor, Martin 111  
Technical University of Berlin 132  
technology 191  
Theory of Economic Development 43  
Threadless 58-59, 74, 86-87, 89, 193  
*Time Magazine*, YOU as person of the year 27, 32  
TMF 58  
Torvalds, Linus 17, 21, 37, 105, 157  
Toyota 67-70, 86, 91, 166, 193  
transparency 21  
Trendwatching.com 57  
trial and error 49  
Tropical Disease Initiative (TDI) 31, 51  
Tylor, Edward 153

## U

UltraSPARC processor 81, 120  
United Layer 101  
*United States versus Causby* 175, 183  
University of Berlin 141  
University of California 23  
University of Kent 138  
University of Kiel 136-137  
University of Maastricht 132, 145  
University of Munich 138  
University of Trieste 69  
UNIX 99, 106, 121  
user experience 191  
user innovation 49  
user participation 48  
user-driven innovation 27-28, 34, 46, 48  
user-led innovation 151

## V

VA Linux Systems 100  
VA Software 100  
Valente, Brian 161  
Valloppillil, Vinud 107  
value creation 46  
Vanhaverbeke, Wim 23, 26  
Villasante, Jesus 134

Virtual Server 97  
virtualization 52-53  
Virtutech 53  
VoIP 103

## W

Walden International 104  
waterfall method 165  
Web 2.0 62, 111  
Weber, Steven 17, 45, 189  
webservices 42, 61, 63, 90, 111  
WebSphere 114, 129  
Weedman, Jeffrey 71  
West, Joel 23  
*Wide Open* (report) 47, 189  
Wijffels, Herman 24  
Wikinews 85  
Wikipedia 31, 84-87, 89-90, 176  
Windows 97, 105, 160  
Windows Vista 37, 159-161  
*Wired* 13, 17, 29  
Wolf, Robert 142  
Wolfe, Bob 68-69  
workstation, virtual ~ 17, 19  
World Design Team (Boeing) 80, 91, 190  
World Wide Web Consortium (W3C) 129  
Wyns, Lode 182, 184-185, 194

## X

Xerox 18  
Ximian 100, 102-103  
XML 111

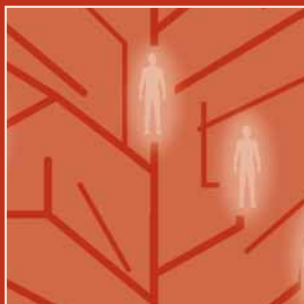
## Y

Yahoo 63-65, 89-90, 185, 193  
Yahoo Maps 185  
Yggdrasil Computing 100  
Yoshimoto, Yutaka 26  
*YOU (Time Magazine)* 27, 32  
YourEncore 72-73, 91  
YouTube 30, 111

## Z

Zeitlyn, David 138, 141  
Zend 36, 102, 112





## Open for business

### Open Source Inspired Innovation

#### About this book

Innovation is the engine of the economy, prosperity and progress. In our era of computers, Internet and smart software, innovational inspiration is encroaching from all directions. In order to exploit the new business opportunities that are becoming available, organizations are being compelled to open themselves up to the masses. This means first and foremost that ideas must be fostered via intense interaction with the outside world: partners, knowledge institutes, competitors, and individuals. Expert users play a crucial role in this dynamic environment. This trend, which started with open source software development, is now clearly spreading through all economic sectors.

*"When we started Brewtopia, we never had a book like this one to tell us how to be 'Open for Business', but I was there at the bleeding edge, developing robust tenets and principles. So I discovered from first-hand experience that, when applied correctly and with diligence, the practices and lessons described in the chapters you have before you will certainly help you in setting up successful open-source and crowd-sourced businesses."*

**Liam Mulhall**

Chairman, Managing Director/CEO of Brewtopia Limited

[www.brewtopia.com](http://www.brewtopia.com)

*"The traditional notion of what an organization is, how it is structured and how it operates, is the equivalent of a silent movie in the 21st century. Human beings are deeply social, and they have an innate need to connect and collaborate. Therefore, understanding the philosophical and structural implications of the open source movement is imperative. Ignore this book at your peril!"*

**Alan Moore**

Founding director of SMLXL

Co-author of Communities Dominate Brands

[www.smlxtralarge.com](http://www.smlxtralarge.com)

#### About VINT

With VINT, the Research Institute for the Analysis of New Technology founded in 1994, Sogeti Group inspires organizations by means of publications, seminars and workshops on the impact IT innovations will have. Recent research topics have been: Service-Oriented Architecture, IT Governance, Innovation, and New Media.

VINT | Vision • Inspiration • Navigation • Trends

ISBN 978-90-75414-19-6



9789075414196



From left to right: Erik, Jaap and Menno