The Structure of Digital Computing

From Mainframes to Big Data

by

Robert L. Grossman

Open Data Press LLC
2012

Published by Open Data Press LLC
400 Lathrop Ave, Suite 90
River Forest, IL 60305, USA

Library of Congress Control Number: 2012908445

Printed in the United States of America

First Printing: June, 2012

ISBN 978-1-936298-00-6

Contents

Preface

This book is about the structure of digital computing: what is significant, what is novel, what endures, and why it is all so confusing. The book tries to balance two point of views: digital computing as viewed from a business perspective, where the focus is on marketing and selling, and digital computing from a more technical perspective, where the focus is on developing new technology.

My goal was to write a short book about digital computing that takes a long term point of view and integrates to some extent these two perspectives.

The book is shaped by my personal experience in these two worlds: From 1996–2001, I was the Founder and the CEO of a company called Magnify, Inc. that developed and marketed software for managing and analyzing big data. Prior to this, from 1988–1996, I was faculty member at the University of Illinois at Chicago (UIC), where I did research on data intensive and distributed computing. From 1996–2010, I remained at UIC as a part time faculty member.

I wrote the sections in this book over an approximately eight year period from 2001 to 2008, with most of the writing done during 2001–2003. I have left the older sections by and large as they were originally written.

Although there have been some changes since 2003 (for example, computers are faster, there are more web sites, and phones are smarter), hopefully as the book will make clear, at a more fundamental level, we are still on the same fifty or so year trajectory today that we were on in 2003.

Robert L. Grossman

Chapter 4

Technology Adoption Cycles

4.1 Forces Effecting Technology Adoption

> Scientists seek to understand what is, while engineers seek to create what never was.
>
> Attributed to Theodore von Karman by Henry Petroski

In the prior chapter, we discussed technical innovations and how it is often hard to recognize them immediately from the clutter in the market. In this chapter, we discuss a complementary question. Given a new technology, what are the basic forces that determine how long it takes for the technology to become accepted by the market? Specifically, we discuss four such forces:

- Technology road maps, which chart the trajectory of a technology as it proceeds from basic research to engineering and manufacturing.
- The context that reflects how the new technology integrates with the existing technological base.

- General market forces affecting the vendors developing and marketing the technology.
- The pain and pleasure of users, which determine in part which technologies they purchase and when.

If you have trouble remembering these four forces, think of the initials RCMP for Roadmap, Context, Market forces, and Pain and pleasure. We begin with a brief overview of these four forces.

Technology Roadmaps. The introduction of innovative technology into the market place generally follows a standard trajectory, beginning with basic research, followed in turn by applied research, engineering development, and ending with manufacturing and marketing. A decade or more can elapse from the beginning to the end of this process. For example, Hotmail, the free email service now owned by Microsoft, was launched on July 4, 1996 and two and half years later in February, 1999, it had over 30 million users [89]. From one viewpoint, a subscription base this large after two and a half years is unprecedented, yet from another viewpoint 1999 is 28 years after 1971, the date the first email was sent by Ray Tomlinson, a computer engineer working for Bolt Beranek and Newman in Cambridge, Massachusetts, and eight years after 1991, the date the first web server was developed by Tim Berners-Lee, a software engineer working at CERN in Geneva. This is sometimes summarized as saying that although it is difficult to predict what technology will look like in twenty years, it is certain that if a technology is not in a lab today, then there is not even a possibility that it will be deployed in twenty years.

The Context. A new technology is generally not used alone but almost always integrated with other technologies and business processes. Think of this as the context. Different technologies can have quite different contexts. One way of determining a technology's context is by understanding how a technology integrates into an organization's IT infrastructure and business practices. For example, many

companies use Customer Relationship Management (CRM) software to reduce the wait time on the phone for a company's best customers and to provide them with higher quality service when working with them. To implement this requires several steps: first, data must be collected about customers; second, the data must be analyzed in order to develop a useful segmentation of customers; third, this segmentation must be integrated with the telephone system; fourth, the customer service representatives in the call center must be trained; fifth, the process must be maintained over time, as new products and systems are introduced. The complexity of the integration limits the adoption of the technology. In contrast, if a user has a web browser and Internet access, a new web based email account, such as Google's Gmail, Microsoft's Hotmail or Yahoo! Mail, can be set up in less than five minutes. This simplicity was one of the reasons that Hotmail was able to build up a user base so quickly. From the perspective of context, one of the reasons that web based email took off so quickly is because it took off in a vacuum compared to other applications, such as back office and front office applications.

Market Forces. It is likely that both web-based email and CRM software will be used in 10 years. On the other hand, it is also likely that many of the vendors providing the software will be different. A variety of market forces determine the fate of technology vendors, many unrelated to the the technology itself. Vendors get acquired, make acquisitions, introduce new product lines, shed old product lines, make $1B revenue errors on their SEC filings, etc. From the viewpoint of a technology user, it is not a bad approximation to assume that the adoption of a new technology is determined by the technology roadmap and by the context. In contrast, the entrance and exit of technology vendors into and out of the market can perhaps best be viewed as a random walk. To say it another way, it is relatively easy to predict that CRM technologies and web based email will enter the market, as well as the velocity

of their adoption, but relatively hard to predict which particular vendors will be marketing these technologies. Businesses pay a lot of money to experts to hear about market forces, but this will not change these facts, it simply makes businesses feel a little bit better.

The Pain and Pleasure of Users. Users usually buy new technology for one of two reasons: it solves a problem by addressing a pain they currently have in their work, such as reducing the time to fill out a travel expense form from fifteen minutes to five minutes, or it provides pleasure, such as an audio hard disk jukebox, which can store 1000 songs and fit into a shirt pocket.

The conventional wisdom is that a good company with a good CEO can get a mediocre product into the market. In contrast, the point of view here is that which technologies enter the market is primarily a factor of the technology trajectory and the context. Given the right trajectory and the right context, a technology will most likely reach the market. In most cases, a market, or more accurately, a market segment, is dominated by three to four vendors. One should perhaps be a bit cautious when drawing conclusions based upon the characteristics of the surviving companies and their CEOs lest one falls victim to post hoc ergo propter hoc.

4.2 The Basic Equation of Marketing

> The chasm is, by any measure, a very bad place to be.
>
> Geoffrey A. Moore, Crossing the Chasm, HarperBusiness, New York, New York, 1991, page 63.

This section introduces a very simple model of technology adoption that is summarized in Tables 4.1 and 4.2. One of the factors in this model is the approximate size of an existing or potential market. One way of estimating this is to multiply the estimated Average Selling Price (ASP)

of the product or service by the estimated number of units sold each year (Unit Sales).

Market ($/year) = Average Selling Price ($/unit) × Units Sold per Year (units/year).

This equation places companies in different cells, as indicated in the Table 4.1. In this table, there are 12 cells spread over three columns (labeled $1M, $10M and $100M) and four rows (labeled $100, $1000, $10,000 and $100,000). Each cell has quite different characteristics and challenges. For example, if a product costs about $1000, then a ten thousand units per year must be sold for the company to have a yearly revenue of about $10 million, the approximate revenue target for a technology company considering entering the public markets before the Internet bubble temporarily lowered the threshold. A company with $1 million has about 10 employees and one level of management. A company with $10 million of revenue has about 100 employees and two to three levels of management. Every time a company doubles or triples in size, it must change the way it does business, and it must change many of its underlying business processes.

If you talk to someone who sells technology products, you quickly realize that some people like to buy products as soon as they enter the market, others like to wait until *someone* else they know has bought the product, others like to wait until *many* people they know have bought the product, while others like to wait until *everyone* else they know has bought the product. One of the things that survived the Internet bubble is catchy names for these categories of users: they are generally called innovators, early adopters, the main street market, and laggards.

From this perspective, many companies tend to fall into one of four groups, as illustrated by the four rows in the Table 4.1. For example, the challenge of a software company selling department level software applications, such as databases, is to get several hundred innovators to adopt their product, followed by several thousand early adopters,

ASP	$1M AR	$10M AR	$100M AR
$100	10,000	100,000	1,000,000
$1,000	1,000	10,000	100,000
$10,000	100	1,000	10,000
$100,000	10	100	1,000
	Innovators	**Early Adopters**	**Main Street**

Table 4.1: It is sometimes helpful to view companies as falling into one of four broad categories, determined by the average selling price (ASP) of their product or service. In the table above, there are separate rows for an ASP of $100, $1,000, $10,000 and $100,000. As companies grow in revenue, they face a number of difficult transitions in their operations and in their marketing. The table above considers three stages: when a company's annual revenue (AR) is $1M, $10M and $100M. The type of customers who buy new technology changes as the technology grows more mature and becomes more familiar. The terms innovators, early adopters, main street, and laggards were introduced in [92] to describe some of these different groups of customers. The middle columns in the table above show the number of units that must be sold each year to reach sales of $1M, $10M and $100M.

and continuing on to the main street market of tens of thousands of customers.

As the table indicates, the target user for each row is different. In technology, individuals usually purchase items that cost a few hundred dollars; small offices or home offices (SOHO) usually purchase items that cost several thousand dollars; small companies or departments within larger companies usually purchase items that cost $10,000, while larger companies (enterprises) usually purchase items that cost $100,000.

Although each of the transitions from one column to the next is difficult and requires a different marketing strategy, the transition from the market of early adopters to the

main street market is generally extremely challenging and has become known as the chasm. The name is from an influential marketing book of the 1990's written by Geoffrey A. Moore called *Crossing the Chasm* [92], which focused on this transition.

ASP	**Innovators**	**Early Adopters**	**Early Majority**	**Late Majority**	**Late Adopters**	**Laggards**	
100	30,000	130,000	340,000	340,000	130,000	30,000	individual
1000	3,000	13,000	34,000	34,000	13,000	3,000	SOHO
10,000	300	1,300	3,400	3,400	1,300	300	dept.
100,000	30	130	340	340	130	30	enterprise
AR	3M	13M	34M	34M	13M	3M	100M
AR/TR	3%	13%	34%	34%	13%	3%	100%
Years	3	2	2	2	2	3	14

Table 4.2: A slightly more sophisticated model of the technology adoption cycle for a product with a total revenue (TR) of $100,000,000 over a lifetime of 14 years in which the annual revenues (AR) follow a bell-like curve. The first four rows of data show the relationship between the average selling price (ASP) and the number of units sold (Units Sold) to reach the annual revenues indicated. The seventh row of the table shows the percentage of the revenue that year compared to the total lifetime revenue. Each market (innovators, early adopters, early majority, etc.) has its own psychology so that there are gaps between the market strategies required for the different segments. The gap between the early adopters and the early majority is sometimes particularly wide, and has become known as the chasm.

We close this section by showing how Tables 4.1 and 4.2 can be used to provide a back of the envelope understanding of the economic constraints on new products. Here is a simple example, sometimes called Bill's Law, after Bill Joy, the Co-Founder and Chief Scientist at Sun Microsystems. Bill (Joy)'s Law: "Don't write software for less than 100,000 platforms." One hundred thousand unit sales at $1000/unit generates $100,000,000 of revenue per year. Ten percent of this is $10,000,000/year, which is enough to support about a hundred engineers. In 2001, Sun did over $18B in revenue and had over 43,000 employees.

Here is another simple example, sometimes called Bill's Law, after Bill Gates, the Chairman and co-founder of Microsoft. Compare Bill (Gate)'s Law: "Don't write software for less than 1,000,000 platforms." One million unit sales at $100/unit generates about $100,000,000 of revenue per year. As before, ten percent of this is $10,000,000/year, which is enough to support about a hundred engineers. In 2001, Microsoft did about $25B in revenue, and had over 47,000 employees.

4.3 Getting to Main Street

> Technology has the shelf life of a banana. By the time you buy it, implement it and train people on it, it's obsolete. The right thing to do is to share IP. Rather than litigate and protect our IP, we've decide to innovate and share it.
>
> Scott McNealy (1954 –), Co-founder and Chairman of Sun Microsystems.

A popular measure of the adoption of technology in the consumer technology market is the number of years it takes a new technology to reach a million users. In "market speak," once this occurs, the technology is said to have reached Main Street. An interesting comparison is provided by the automotive industry. The first automobiles were introduced in about 1902 and by 1914 there were over

a million registered automobiles, which was about 1% of the US population at that time. While AOL cost about $20/month and Hotmail was free, an automobile in 1914 cost about $2000, which is about $36,000 in today's dollars.

One way of looking at Main Street is in terms of market penetration. Think of the approximate size of the US market for consumer goods and services as about 300 million individuals. Then a user base of 3 million is equivalent to a market penetration of about 1%. Since the size of markets can be hard to estimate, knowing the absolute number of new users each year and total user base is an alternative metric that is also frequently used.

Companies take a variety of paths, usually involving many detours, to reach Main Street. Everyone likes winners: winners tend to be written about and remembered, while those coming in second or third tend to be ignored. A famous example of a technology company that reached Main Street fairly quickly is Sun Microsystems. It reached Main Street in just six years after it was founded in 1982. By 1988, Sun reached revenues of $1B/year by selling approximately 100,000 computer workstations at about ten thousand dollars per unit [137]. There are a number of things to be cautious about when viewing technology from this perspective. Perhaps the most important is to be aware that there is usually a significant gap between the introduction of a new technology into the market place and the emergence of a company that carries the technology to Main Street. Usually several vendors enter and leave the market before one any of them reach Main Street. It is common to view this as a failure to market correctly; on the other hand, it is perhaps better viewed as a mistake in entering the market too early. The trajectory of a technology as it follows a technology road map is usually quite long, often time taking a decade or two; predicting the general time period within a few years of when a technology is ready to reach Main Street is fairly easy; predicting the particular year is much harder. Unfortunately, it is the latter which

is often a critical success factor for a technology vendor.

From a marketing perspective, reaching Main Street is critical, since as can be seen from Table 4.2, most of a company's revenues (and, it is hoped profits) come during this period. For the hypothetical company in the table that has a product in the market for 14 years, 68% of the revenues come during the 4 years it is on Main Street. On the other hand, most of the development costs occur during the preceding 5 years, and these development years are often unprofitable.

For successful technologies, although several companies will reach Main Street, predicting which companies reach it is generally not easy. An important factor is when they enter the market. The closer the entry point to the inflection point in the roadmap, the more likely the company is to reach Main Street. Companies entering too early often don't have the staying power; companies entering too late often don't have the marketing power to displace the current market leaders.

Analyzing products and vendors from this point of view is complicated by several factors: using the hypothetical example above covering the 14 year history of a product ignores the fact that over this period, a product undergoes a number of significant changes and it is perhaps simply a matter of convenience whether to consider these as evolutions of a single product or as genuinely new products.

Another complicating factor is that most companies that have a product on Main Street also sell and market a number of other products. Each of these other products may be building towards Main Street, on Main Street, or transitioning off Main Street.

We conclude this section with a final observation. Just as there are two views of the stock market — some feel that the prices of individual stocks cannot be predicted, with the fluctuations in prices most accurately modeled as a random walk, while others feel that the behavior of individual stocks can be predicted by experts — there are two views of vendors.

One view of vendors, championed by the analysts who track them, is that with a proper knowledge of the space, one can predict which vendors will prosper, which will fail, and which will limp along. Another view, less common, is to model the progress of vendors as a random walk.

Viewing vendors in this way highlights an important factor complicating the adoption of new technology — the lack of stability of the vendors that supply new technology due to market forces (the "M" in the acronym RCMP introduced at the beginning of this chapter).

Despite these complications, the basic facts are clear, and it is useful to summarize them here again: Predicting technology breakthroughs is difficult; identifying technology breakthroughs is much easier. Predicting which particular vendors will reach Main Street is difficult; predicting technology trajectories based upon technology road maps is much easier. From this perspective, understanding the adoption of a new technology is relatively easy to understand, whether it is an automobile at the beginning of the last century or the Internet at the end of the last century.

4.4 Case Study: The Nike Pegasus

The adoption of any new technology, including the technology in running shoes, depends upon a complex interaction between the usefulness of the technology, the savvy of the marketing, and the receptivity of the user base. New technology creates opportunities, but without the push provided by innovative marketing and the pull of a cohesive user base, technology is not adopted but rather simply evaporates over time.

Nike's shoes are branded with a swoosh and were endorsed by Michael Jordan. The company was started in 1964 and by 1987 had developed four generations of the Air Pegasus and sold five million of them. Nike is generally considered to be a marketing success. On closer examination, it is just as noteworthy as a technology success. In

Year	Production	Registration
1900	4,192	8,000
1902	9,000	23,000
1904	22,130	54,590
1906	33,200	105,900
1908	63,500	194,400
1910	181,000	458,377
1912	356,000	901,596
1914	548,139	1,664,003
1916	1,525,578	3,367,889

Table 4.3: The mass production of automobiles started in 1908 with the Model T Ford. Within six years, over one million automobiles had been registered. Within eight years, over one million automobiles were being produced a year. Source: Facts and Figures, Automobile Manufacturers Association, New York, New York, 1950. Also, Encyclopedia Britannica, Motor Cars, Volume 15, page 881, Encyclopedia Britannica, Inc., Chicago, 1957.

fact, it is a company that provides an example of a rare combination of marketing and technology innovation.

Bill Bowerman was one of the co-founders of the company, which was originally called Blue Ribbon Sports or BRS. He was the University of Oregon track coach from 1944-1972, and while he was there, his teams won four NCAA team championships and he coached 44 All Americans and 19 Olympians. He observed that every ounce shaved from a shoe decreased by 200 pounds the weight lifted by his runners in a typical race [105]. In 1967, Bowerman began development of BRS's Marathon, which was one of the first running shoes made from lightweight nylon.

The year 1971 was a pivotal one for the company, both from a marketing and a technological viewpoint. In 1971, according to company lore, Jeff Johnson, the company's first employee, had a dream the night before the first shoe

boxes were to be printed. The dream was about Nike, the Greek goddess of victory, and created the identity for the company. Phil Knight, the other co-founder of the company, wanted to call the company Dimension 6, a name that was no doubt too technical to make it a main stream marketing success.

In the same year 1971, Bowerman, while eating waffles at breakfast, got the inspiration for what later became Nike's waffle sole. He prototyped the sole by pouring urethane rubber into his family's waffle iron and then worked hard for the next several years perfecting it. By 1974, the Nike's Waffle trainer had become the best selling training shoes in the United Sates, and his wife had forgiven him for ruining the waffle iron.

When reading accounts about marketing, it is common to hear about inspirations from dreams, eureka moments while taking showers, and defining moments at critical PowerPoint presentations, since these are all interesting to read about. In contrast, technological innovation often seems to come from hard work and good experiments, but this doesn't seem to make for an exciting story. So in general, a casual reader is left with the impression that the adoption of a technology is due mostly to inspired marketing and doesn't hear about hard technical work, unless it makes for a good copy, such as Bowerman's use of the waffle iron. Successful companies seem to have a fair share of both.

In 1979, Nike introduced a new way to cushion shoes: it sealed gas inside polyurethane capsules and embedded the capsules in the shoe's sole. The idea came from Frank Rudy, an aerospace engineer from North American Rockwell, who presented the idea to Nike in 1978. Prior to the use of air cushioning, running shoes used sponge rubber wedges for cushioning. The sponge rubber was relatively heavy and tended to compact over time. In contrast, the polyurethane capsules recover their original shape after impact and do not deteriorate over time as quickly as sponge rubber. One of the reasons that polyurethane capsules work is that the gas in the capsules contains large molecules, so

that relatively little gas escapes from the capsules [105].

During the 26.2 miles of a marathon, a runner's shoes endure more than 25,000 impacts - the use of nylon uppers, waffle soles, and urethane capsules provides an important competitive advantage over traditional materials and construction. Nike began by targeting competitive racers, a relatively small and close knit community, which was receptive to these types of technical innovations. Moreover, Bill Bowerman's status and success as a coach increased the receptivity of the early adopters of these types of shoes.

During the same period of technological innovation, Nike had an equally impressive record of marketing achievements, including the following: in 1973, American record holder Steve Prefontaine was the first prominent athlete to wear Nike shoes. In 1977, the company began a marketing campaign with the tag line "There is no finish line." In 1978, BRS changed its name to Nike and tennis star John McEnroe was featured in Nike's marketing campaigns. In 1984 at the Los Angeles Olympics, Carl Lewis won four gold medals and Joan Benoit won the first women's marathon and both wore Nike shoes. In 1985, Chicago Bulls' Michael Jordan endorsed a line of AIR JORDAN shoes and apparel. In 1988, Nike began its "Just Do It" campaign. In 1992, the "Hare Jordan" Superbowl add featured Michael Jordan playing basket ball with Bugs Bunny [105].

These three forces — technology innovation, marketing innovation, and user receptivity — resulted in the sale of more than five million Nike Pegasus shoes, and are exactly the same forces that result in the acceptance of a new database or operating system.

4.5 Technology Roadmaps

> I don't know what technology will look like in twenty years, but if it is not in a laboratory today, then it is not a possibility.
>
> Traditional

A *technology roadmap* is a series of concrete predictions by an individual or group about the evolution and trajectory of a technology. Technology roadmaps turn out to be quite handy due to the long time often required for basic research to reach end users in the market as new technology.

We begin by describing four important phases that most new technology goes through, beginning with basic research and ending with manufacturing and marketing.

Phase 1. Basic Research. The first phase is scientific and consists of the discovery of new principles and phenomena. The goal of basic research is to improve our understanding of basic scientific and engineering principles, structures and mechanisms. A good example from the last chapter is packet switched communication networks that divide messages into packets and route the packets individually, instead of relying on a fixed circuit from the sender to the receiver as was previously done. Another example was the exploration of the basic properties of the elements Gallium and Arsensic and compounds synthesized from them prior to their use in integrated circuits. Today, most basic research is sponsored by the federal government and takes place at universities. Basic research is also done by a few of the largest companies, such as IBM and Microsoft.

Phase 2. Applied Research. Applied research fleshes out the ideas discovered in basic research with the goal of developing new products and services. There is not a sharp division between basic and applied research. Today, there is a lot of research interest in understanding how a quantum computer might work. Most of this is basic research and focused on understanding the underlying principles of a quantum computer. Some very simple quantum devices have been built that can compute and store just a few quantum bits. Over time as the basic principles of building quantum computing devices are better understood, attention will shift to applied research focused on developing quantum devices that can store and compute with enough

bits to be useful in practice. Applied research is done by researchers at universities and by research labs in large companies. In some industries — for example bioinformatics — applied research is also done by smaller companies.

Phase 3. Engineering Development. In the third phase, the ideas and principles discovered in basic and applied research are applied to build new devices and to deliver new services. After something new is discovered or understood, it can take quite some time to learn how to build it efficiently. In many fields, the main way of learning this is to build many wrong versions, until over time, someone eventually stumbles upon a correct version. As a simple example, it usually takes about a year to release a new version of a software product, and usually several versions before the product is mature enough that it is useful to a broad market. Here is a thought experiment to understand how difficult it can be to design and build a product (this only works for people over 40). Think about how many heavy suitcases you lugged through airports and compare that to how easy it is now to pull a suitcase with wheels through an airport today. Now try to understand why you never thought about adding wheels to suitcases. Engineering development is done by companies based upon basic and applied research done either in-house or by others. Small companies are particularly important here.

Phase 4. Manufacturing and Marketing. In the fourth phase, the products and services must be sold in large enough quantities and at good enough margins to create a successful commercial enterprise. Marketing is done by companies to help sell the products. Good marketing is quite hard. As we learned in the last chapter, marketing in general, whether good or bad, generally leads to clutter in the market. The products that companies market and sell can be either manufactured by themselves or outsourced to others to manufacture.

Here is an extreme simplification of these four phases: scientists discover things, applied scientists refine these dis-

coveries, engineers build things, and marketers explain to people why they need things.

As a simple example of these four phases, consider the development of the web server and web browser. The basic computer science research underlying the browser included advances supporting packet switched networks, distributed computing, and related areas. As web servers and browsers became widely deployed, applied research was undertaken in a number of areas in order to develop more robust and scalable web servers and web clients. For example, techniques were developed so that a single web site could share its work load across dozens or hundreds of web servers. By sharing the web load in this way, NASA web sites do not crash when new images from Mars become available, IRS web sites do not crash around April 15, and Victoria's Secret web sites do not crash when they stream online webcasts.

The first browser, called Mosaic, was developed by a small research group at the University of Illinois at Urbana-Champaign. As the number of users began to grow, there was commercial interest in browsers and engineering development was done by Netscape and Microsoft to produce browsers with improved stability, performance and functionality. Finally, the "browser wars" between Netscape and Microsoft were part of Phase 4, the marketing and manufacturing of the browsers.

One of the better known technology roadmaps is the yearly International Technology Roadmap for Semiconductors or ITRS. The 2005 ITRS roadmap contained over 200 tables of data and made thoughtful predictions about semiconductors 15 years into the future. The ITRS is sponsored by several semiconductor industry associations from around the world and every two years identifies technology challenges and needs facing its members for the next 15 years. The 2001 ITRS roadmap was developed by over 800 experts from around the world organized into various working groups and committees. For example, the 2001 roadmap predicted that microprocessors speeds will reach

over 10 GHZ by 2010 and memory chips will be 8 GB. It is important to note that it doesn't predict which vendors will still be in the business of producing and marketing memory chips in 2010. Table 4.5 contains some of the key findings from the 2001 ITRS.

4.6 Case Study: Clusters

A good example of the decades required to roll out a new technology is provided by what is called cluster computing. The basic idea is simple: lots of small commodity computers are linked together via a network to create a virtual supercomputer. With the commoditization of hardware and networking, this has become a cost effective way of creating supercomputers that are quite effective for certain tasks.

This idea is familiar to many from the SETI@home Project, which uses idle time on personal computers to search for extraterrestrial intelligence. The software to do this was made available in May, 1999. By July, 2002 over 3.8 million individuals had downloaded the software and donated some of the unused time on their computer to the project. With this strategy, the team was able to build one of the top ten most powerful supercomputers at the cost of a few servers and the cooperation of a few million personal computer owners who were interested in whether they are alone in the universe.

Here is another example. On January 8, 2003, a USA Today article with the title "High tech's latest bright idea: Shared computing" explained another application of this technology to the search for new drugs in this way:

> By day, Richard Vissa's PC sends e-mail, lets him surf the Web and handles other ordinary tasks. By night, when Vissa goes home, it becomes a super-powered computing machine, testing potential drugs for their ability to defeat disease. ... The company is one of many dabbling in a new technology that many in tech,

> including IBM, say is one of the most promising new technologies in years. Dubbed shared computing, the technology allows companies to harness the processing power in every computer at all times and to combine it to crunch big computing tasks that before required expensive supercomputers.

There is a problem here, and it involves USA Today's use of the word "latest" in the phrase "high tech's latest bright idea." There is an implicit requirement imposed upon technology writers that they write articles about breakthrough ideas. This is because newspapers and magazines don't usually publish articles unless they involve a "breakthrough" or the "latest bright idea." This is despite the fact that all good technology writers know that technology can take years to be adopted by the marketplace. For example, more than eight years before the USA Today article, Business Week reported about a project that I was involved with using almost exactly the same words:

> Have a major problem that demands a supercomputer, but limited funds? ... Build a "virtual supercomputer" ... [by linking] a bunch of workstations...

In the 1993 grant proposal to the National Science Foundation to fund this project, a variety of prior and related research was cited stretching back to the late 1970's. There were two basic ideas: The first idea was that a bunch of computers in one location on a common local area network could work together on a common task. This is the idea of a cluster computing. Later, instead of a standard local area network, other more specialized techniques were also used to connect the computers and form a cluster.

The second idea was that geographically distributed computers from different administrative domains could also be linked together to create clusters of clusters or what were

sometimes called meta-clusters. About five years later, in 1999, the idea of meta-clusters began to take off in a serious way as several supercomputing centers began to link their supercomputers together in what became known at that time as grids.

There were two types of clusters deployed. The first type were dedicated clusters that were devoted entirely to forming a virtual supercomputer. There were also clusters that used what was sometimes called "cycle stealing." With cycle stealing, computers could also be used for individuals for their standard work, and either at night when the computers were not being used, or at other times when the computers were idle, the computers would be incorporated automatically into the virtual supercomputer.

Several start ups tried to commercialize cluster computing in the early 1990's without much success. Several years later, in the late 1990's several companies tried to commercialize grid computing without much better luck. On the other hand, a company called Platform Computing that was around with related technology in the early 1990's has found a nice niche marketing cluster and grid technology, and, some of the larger companies such as IBM, Sun, and HP incorporate cluster and grid technology in their offerings.

More recently, cluster and grid computing have been reinvigorated by what is called cloud computing. With one type of cloud computing (sometimes called hosted clouds), clusters at the scale of data centers are operated by third parties, and computing is provided as a utility in the sense that you can get as little or as much as you need, and you pay for it by the slice. This greatly simplifies the delivery of cluster-based computing services and has made it easier for companies to adopt this technology.

Cloud computing was the cover story for the December 13, 2007 issue of Business Week and was described as "[as] a fundamental shift in how we handle information" [7], and in a later article by Business Week as "a major shift in the way companies obtain software and computing capacity [71]."

To summarize, the adoption of cluster computing by the marketplace took about two decades and went through several name changes and a few different business models. Throughout this process, vendors supplying cluster technology entered and left the marketplace. The adoption of cluster technology is fairly typical. It is usually quite difficult to predict *which vendors* supplying a technology will survive, but relatively easy to predict *which technologies* will survive.

4.7 Context

Context is one of the four basic forces introduced in Section 4.1 determining how long a new technology takes to reach Main Street (the "C" in RCMP). Context from this viewpoint includes three different but related aspects:

- Complexity. Some technologies are intrinsically more complex than others. Sending a message by email is simply not as complex as analyzing the digital data produced by a million customers and then using this analysis to improve a company's direct marketing. Also, the more complex a technology, the more decision makers in an organization are involved, and hence the more complex the sales cycle.

- Lock-in. Lock-in describes strategies employed by vendors to make it more difficult to switch to the technology offered by another vendor, as well as the implicit lock-in provided by a large base of users, who are generally reluctant to switch to another technology, even if it is better.

- Standards. Over the years, standards have played a more and more important role in the commoditization of technology. New technologies that leverage existing and emerging standards tend to be adopted more quickly.

In this section, we will describe each of the aspects of context in more detail, beginning with complexity.

Complexity. Perhaps the simplest way to think about complexity is that it is the underlying reason that so many technology projects fail. Here is my list of the top three ways in which complexity causes to technology projects to fail.

Reason 1 - It's is the people. The first reason is that a lot digital computing is simply too complicated for most people to use. Making the situation worse is the fact that most people who develop technology are pretty happy with the technology they develop, but pretty disappointed in the people who use it. The reason is that technology must be extremely well designed in order for people to use it effectively, and, unfortunately, most technology is not this well designed. Given the choice of blaming the technology or blaming the users, most developers find it much easier to blame the users and not to change the technology.

Reason 2 - It's the project. Deploying a new technology is a complex project. All complex projects are challenging for reasons having nothing to do with technology. Understanding this is as simple as thinking back to the last time you went to a large dinner party. Usually, there is a discussion about the problems someone is having remodeling their kitchen. The project is inevitably behind schedule. Moreover, there are always problems with the contractors and there are always surprises.

But think for a moment about this. Most people who remodel their kitchen have used an earlier kitchen for many years, have spent lots of time in other peoples' kitchens, have had many discussions with their friends about their friends' problems remodeling their kitchens, and then act surprised that their particular remodeling project is behind schedule, over budget, and likely to be disappointing when completed.

Compare this to a technical project. Many technical projects provide new capabilities and those running the

project often have little prior experience that is directly relevant. At the same time, technology projects share all the structural and organizational characteristics of any complex project, such as remodeling a kitchen. One should not be surprised that technology projects end up behind schedule, over budget, and with disappointing outcomes, just like remodeling a kitchen.

Reason 3 - It's the integration. Typically an IT system has been built over a period of years, by a number of people, through a large number of iterations. In general, no one likes the system. The only thing it has going for it is that it works. This is not to be underestimated. Its replacement is likely not to work.

The reason it works is sometimes a bit subtle. Through a series of iterations over years, the system gradually becomes better and better integrated with upstream systems and processes that feed it and with downstream systems and processes that it feeds. Say the system is one that processes direct mail campaigns and that it was built ten years ago. Assume that it takes an hour to prepare a campaign and the response rate of the campaign is 2.5% on average. Assume that the new system promises to prepare campaigns in 20 minutes and to get response rates of 3.0%.

The difficulty is the work required to integrate the new system into the organization. The correct data must be fed in, and the outputs must be integrated with the required downstream systems. The prior work that integrated the old system into the required upstream and downstream systems is likely to have been done by folks who have long ago left the organization. The current staff often have not done an integration like this before and are already behind schedule in other projects that are much less challenging and threatening. The result is what you would expect: many technology projects fail.

Lock-In or the Tyranny of Vendors and Users. One of the more interesting battles in technology is the battle between vendors to lock their customers into a technology

and thereby guarantee a long-term revenue stream, and their customers' desire to be able to switch vendors and technologies.

To phrase it more dramatically, the emergence of a new, and sometimes better, technology is thwarted by both the tyranny of the installed user base, who generally just want to be left alone, and the greed of the vendors, who are pursuing lock-in strategies designed to make it difficult for a user to switch a technology once it is deployed.

Once a technology reaches Main Street, it is usually so inexpensive that it lingers for quite a while, even if new technologies are much more efficient. For example, the Internet still has not eliminated the fax. As Odlyzko notes, "efficiency often plays a minor role when relatively inexpensive goods or services are involved [116]."

Contributing to lock-in is the cost of switching to a new technology, which can be very expensive. Here are some of the costs that may be involved:

- Search costs. Finding a new technology can often involve direct costs for searching for the technology. Most companies cannot switch to a new technology without a very large number of meetings. Just the coffee and donuts for the meetings begin to add up after a while. Complex projects usually involve pilot projects, which themselves can sometimes be expensive.

- Contractual commitments. Vendors try to negotiate long-term contracts. Getting out of a contract can be expensive.

- Capital costs. New technology can require new equipment. As a simple example, for many home users a new version of Microsoft Windows generally requires either a new computer or many free weekends and lots of coffee.

- Training. Learning new systems involves direct costs

for training and indirect costs for lost productivity during the transition period.

- Migrating data. Moving data from one format to another and from one system to another can be very expensive.

- Integrating new suppliers. Transition to new technologies sometimes requires new suppliers. Integrating new suppliers can sometimes be complex.

Standards and the Hope They Provide. When someone hears the word "standard" today, they often think of software standards such as HTML, HTTP, XML, etc. Standards, though, have always played a critical role in the adoption of new technology. A good way to understand the importance of information technology standards is to look at automotive standards and railroad standards.

The first Ford Model T was produced in 1908. Prior to that, cars were custom built by hand for the wealthy. In the first year of production, more than 10,000 model T's were sold for $950 each. In 1914, Ford sold more than 308,000 cars, which was more than the other 290 some automobile manufacturers combined. In 1915, he lowered the price to $280 and sold a million cars. The first Ford Model T took over 12 hours to assemble. By 1913, the time had been reduced to 1 hour and 33 minutes [19].

Ford introduced the production line to lower the cost of assembling cars. The Ford production line for the Model T consisted of a moving conveyor belt and different stations, where workmen repeated the same assembly task. This required parts which were precisely manufactured according to standards and were interchangeable.

To illustrate this approach, Henry M. Leland, the founder of Cadillac Motors, organized a demonstration of the British Royal Automobile Club in 1908. In the demonstration, three Cadillac cars were disassembled and the parts were mixed together. Then 89 parts were removed at random and replaced with parts from inventory. The cars were

then were assembled and driven 500 miles without a problem [19].

The same basic ideas are at the root of the commoditization of the mainframe computer during the first era of computing and the PC during the second era of computing. Today, a PC can be assembled relatively easily by putting together standard motherboards, CPUs, disk drives, memory, and an operating system.

In contrast, the third era of computing is essentially controlled by communication standards, which can be viewed as broadly analogous to railroad standards. The gauge of a railroad is the distance between the inside faces of the two track. The most common gauge today is 4 feet 8 1/2 inches or 1435 millimeters, which was introduced sometime between 1822 and 1825 [24]. Without a standard, railroad service would be fragmented and limited to tracks of a single operating entity. The same is true of communication networks. The more networks that are interconnected, the more interesting and useful the network. This depends upon running a common set of communication protocols, such as the TCP/IP protocols described in Chapter 2.

The standards associated with the fourth era of computing are somewhat different and perhaps best thought of as broadly analogous to the Generally Accepted Accounting Practices (GAAP) used by accountants. In theory, two different accountants faced with the same books of a company and working independently and both following GAAP would come up with the same balance sheet, income statement, and cash flow statement. Of course, in practice this is not the case, but that is the goal. Web services are similar.

4.8 Case Study: Relational Databases

The technology adoption cycle can be seen quite clearly in the adoption of relational databases. A relational database is a software application that views data as a collection of rows and columns and provides mechanisms for creating ta-

bles of data, querying tables of data, and updating tables of data. Today the majority of business data is stored in relational databases, and, by and large, business are better off doing this than not doing this. The adoption of relational databases, though, was a process that took several decades.

At the beginning of the 1960's, many mainframe business applications could be viewed as processing data records by applying certain business rules. For example, a payroll application might read a record containing the number of hours an employee worked, read another record containing the hourly rate for the employee, read another record containing the employee's department, and use this information to prepare a check for the employee and address it to his department for delivery. The business rules would specify that the number of hours should be multiplied by the hourly rate to determine the gross payroll and then specify how to calculate the size of the deductions for federal taxes, state taxes, local taxes, health benefits, retirement deductions, etc. For large companies, this was a major advance over manual systems for payroll.

If an employee was promoted and moved to another department, one record containing the employee's hourly rate and one record containing the employee's department could be updated and the next payroll run would create a pay check with the correct salary and deliver it to the correct department.

As more and more payroll and similar back office systems were built, software engineers began to realize that the software code for working with records had certain common functions, including: 1) functions for creating tables; 2) functions for reading tables; 3) functions for updating tables; 4) functions for joining two tables together to produce a third table. These properties became the core of a relational database.

The conceptual framework for databases based upon tables containing data records and using joins to produce richer tables was described in a seminal paper by Edgar

(Ted) Codd with the title "A Relational Model of Data for Large Shared Data Banks," which appeared in 1970. Databases following the principles described in the paper became known as relational databases. Codd worked at IBM Research in Almaden, California at the time.

In the early 1970's, basic research and applied research was done in this area by two groups: the Ingres Project at Berkeley and the System R project at IBM. The Ingres Project resulted directly or indirectly in several commercial databases including Sybase's DBMS, Microsoft's SQL Server, Tandem's NonStop SQL, and Informix's DMBS. The System R project at IBM led to IBM's DB2 database.

Prior to relational databases, business data was stored either in custom applications or in what were called hierarchical or navigational databases. Querying these earlier types of databases required a skilled programmer, several lines of code, and knowledge of how the data was physically laid out on the disk. Later, relational databases came along with a language called SQL. Querying data could be done with a single statement in SQL.

The difference between navigational and relational databases can be thought of in the following way: with navigational databases you have to know where the data is in the same way that you have to know where the data is on your hard disk if you want to point and click through a sequence of folders to get there. With relational databases, you can access data by providing certain information to a database query in the same way that with a search engine you can access documents by providing key words to a search engine query. There is an important difference though: search engine queries are easy to write, while database queries are much harder to write.

Although writing SQL queries is complex, it is significantly easier than writing queries for hierarchical and navigational databases. In the 1990's, graphical front ends to relational databases became common, making it even easier for casual users to extract information from databases. Developing a query language with this type of power was one

of the main motivations for the IBM researchers working on what became System R.

In the early 1980's, first generation vendors of relational databases included Ingres (a commercialization of the project at Berkeley), Sybase and Britton-Lee. The conventional wisdom of industry pundits was that relational databases were of interest to academics and useful for certain niche applications, but navigational databases, which were more scalable and had higher performance, were better suited for most serious commercial applications. Pundits recommended using an older technology called data dictionaries instead of relational databases as being the best way to integrate data into applications. During this period, twenty to thirty database transactions per second were possible on IBM 3084 processors [10]

By the beginning of the 1990's, twenty years after Codd's paper, the commercialization of relational databases was well along: relational databases could support hundreds of transactions per second and work with gigabytes of data. There were a variety of vendors providing relational databases including Oracle, IBM, and Sybase. Applied research began to shift to other topics, for example developing more specialized databases, such as databases for spatial data, image data, and scientific and engineering data.

In 2001, thirty years after Codd's paper, Yahoo replaced a proprietary home built system for assembling content such as headlines, stock charts and insider trading on its financial web pages, with MySQL, an open source relational database [75].

This case study illustrates some of the main themes in this book. First, the adoption of a new technology usually takes several decades. In this case, there was a span of almost thirty years between the first paper describing relational databases and their commoditization by open source relational databases such as MySQL. Second, among experts there was general agreement relatively quickly that relational database technology was an important core technology. Third, predicting which vendor of this technology

would survive was beyond the pundits. Of the early vendors of database technology, only IBM has remained a significant market force. Fourth, in this span of thirty years, there has been plenty of clutter: There have been a few, but relatively few new ideas about relational databases, but there have been hundreds of academic papers about them, and thousands of more popular articles.

4.9 The Pain and Pleasure of Technology

A good technology salesperson always tries to understand the difficulties or, in sales jargon, the pain that potential customers feel. This is one of the topics discussed in this section and is the "P" in RCMP. To ease pain caused by current technology is one of the main reasons people buy new technology.

Trying to understand why people buy technology may seem hard. During the year you are reading this, there will be millions of decisions made resulting in the purchase of technology. In principle, each individual is unique and each decision to purchase is unique; therefore, it would seem that predicting the adoption rate of new technology would be difficult.

In actual fact, probably the opposite is true, and people purchase new technology for only a handful of reasons:

As a toy. Toys bring pleasure and people like pleasure. Many decision-makers in technical positions are particularly susceptible to the pleasure of buying and playing with new technology. It is often the reason that they worked so hard to get and to keep their job. The more secure their position, the more likely they are to tell themselves that it is their *responsibility* to try new technology. It is their duty.

To spend a budget. New projects have budgets and budgets get spent. Technology gets bought whether it is

needed or not and whether it works or not. One of the interesting consequences of the Internet bubble was the following. Many new companies were formed and new capital was infused. New projects got started and technology got bought. The pace was such that often when visiting a start-up you would look around and see software costing tens of thousands of dollars that hadn't even been taken out of the box. After the Internet bubble, budgets grew tighter and it was rare to see unopened software around. On the other hand, the same philosophy can often be seen today in marketing budgets. Marketing budgets are designed to be spent, regardless of whether the company has the discipline to measure accurately the return on the dollars spent. Of course it is not always easy to measure the return of marketing dollars: there is a fair amount of truth in the adage that half an advertising budget is just wasted - the problem is you never know which half.

To ease pain. In most situations when someone is in pain, your response is one of sympathy. Not so in technical sales when pain indicates an opportunity. Pain in this situation means something is not getting done and someone is in trouble. For example, customers are complaining that x doesn't work. Or the sales manager doesn't know how many widgets are getting sold and who is buying them. Or the CEO doesn't have nice colorful reports about sales by region and by product to pass to the board who will file them with only a glance. The role of a technology company is to create products and services which, in the ideal world, result in the easing of these types of pain.

The early market is dominated by decision-makers who primarily buy technology for the pleasure it creates. One of the mysterious facts about the world, which perhaps is best not to think too deeply about, is that occasionally new technology works beyond all expectations, and provides a fundamental advantage to those using it. What this means is that, from time to time, adopters of technology in the early market gain a fundamental competitive advantage

over their competitors by deploying a new technology.

This is very similar to winning the lottery. Although only one person wins a lottery, the word spreads extremely quickly and everyone who has bought a ticket feels that his or her purchase is justified. The organization running the lottery is highly motivated to spread the word. The media picks up the story since their audience is anxious to hear this type of story, in order to justify their purchases of tickets.

Technology success stories in the early market are the same. Stories about the (rare) successes of new technology in early markets are widely reported since the technology media realize that this is exactly the type of stories their readers want.

Before we end the section, it is useful to describe two other reasons people make purchases.

To gain a competitive advantage. Well disciplined organizations purchase technology to gain a competitive advantage. Although you would expect this to be a common reason, there are several reasons it is not. First, it is difficult to choose the right time to deploy new technology. The earlier a new technology is chosen, the more likely it is to lead to competitive advantage (since your competitors have not yet adopted it), but the more likely it is to fail (since the technology is still immature). Since it is difficult at most organizations to be associated with projects that may fail, in general managers push out the adoption of technology until it is quite likely to succeed, but less likely to lead to a competitive advantage.

As an aside, most arguments justifying competitive advantage use quadrant diagrams (see Figure 4.1 for an example). For those who haven't used them, this is how a quadrant argument goes. The "good" quadrant is the upper right and with the right technology a company can move from one of the "bad" quadrants to the upper right. Unfortunately, the technology usually doesn't understand that this is why it was being deployed and often does not

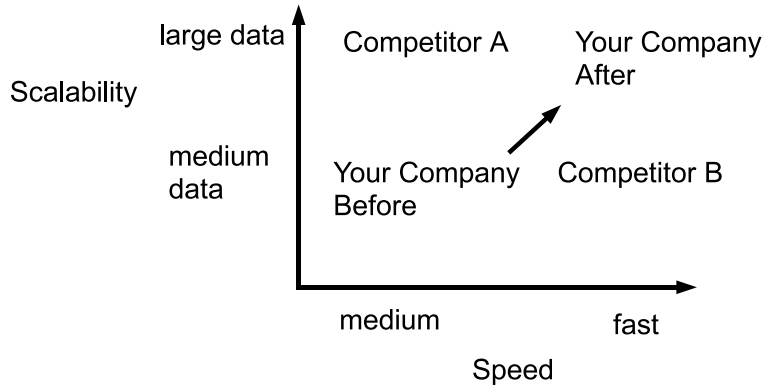

Figure 4.1: The argument for moving to a new technology as prepared by the Director of Marketing of a high tech company. After the technology is deployed, the company will be in the upper right quadrant, the place all high tech companies want to be. One of the jobs of the marketing department is to select two dimensions (speed and scalability in this example) so that the company ends up in the upper right quadrant.

immediately drive the company to the upper right hand quadrant. As a consequence, project managers deploying new technology sometime become demoralized.

As a source of hope. At the beginning of every January, fitness centers are always full and overflowing. By the beginning of February, they are back to their normal levels. The reason is hope. And this is a good reason. For the same reason, many nice, fancy workout clothes are bought each year over the holidays and most are worn only once or twice. Technology is bought for the same reason. Much of it is never installed. Hope is good though. Without hope, we would all go back to paper and pencil.

In the end, people make technology purchases and start technology projects for reasons that are usually not as rational as they may seem at first. Marketers know this and design campaigns exploiting this.

Product	**Number of Years**
Telephones	20
Television	15
Cable Television	10
Cell Phones - Nokia	5
Personal Computers - IBM	< 3
Internet - AOL	< 3

Table 4.4: This table shows the number of years that various technologies took to reach Main Street. It is important to note that most companies never reach Main Street. This is illustrative of the more general phenomenon that while predicting breakthroughs is extremely difficult, identifying breakthroughs is much easier, as is predicting their trajectories, including when they will reach Main Street. On the other hand, predicting which particular vendors will reach Main Street is quite hard, and perhaps best modeled as a random walk. The table is somewhat misleading: in some sense, most of these technologies took about 10-20 years to reach Main Street when viewed from the perspective of the entire technology road map from laboratory to end user. On the other hand, the time for a particular vendor to reach 1 Million end users depends very strongly on when they entered the market. For example, AOL reached a million users less than 3 years after it was started in 1985; on the other hand, that was over 13 years after the first email was sent in 1972.

	2001	**2005**	**2010**	**2016**
DRAM Half-Pitch (nanometers)	130	80	45	22
DRAM Memory Size (megabits or gigabits)	512M	2G	8G	64G
DRAM Cost/Bit (microcents)	7.7	1.9	0.34	0.042
Microprocessor Physical Gate Length (nanometers)	65	32	18	9
Microprocessor Speeds (MHz)	1,684	5,173	11,511	28,751

Table 4.5: This table contains some technology predictions made by the International Technology Roadmap of Semiconductors (ITRS) in 2001. They predicted that by 2010 the cost of DRAM memory will decline to 1/20 of the cost in 2001 and microprocessors will be 10 times faster. By 2016, the cost of DRAM memory will be less than 1/100 of the cost in 2001 and microprocessors will be 15 times faster.

Year	Event
1961	Charles Bachman at the General Electric Company develops the Integrated Data Store (IDS), an early database.
1967	The CODASYL Database Task Group (DBTG) begins work on the Network Database Model.
1968	IBM introduces a database called IMS running on IBM System/360's. IMS uses a navigational data model, but is not compliant with CODASYL data models being developed at the time.
1970	IBM's E. F. Codd publishes the first paper describing relational data models.
1971	The Conference on Data Systems Languages (CODASYL) releases its first standards report. CODASYL products are developed by Eckert-Mauchly Computer Corporation, Honywell Incorporated, Siemens AG, Digital Equipment Corporation (DEC), and Prime Computer Corporation.
1973	Another prototype relational database project called Project Ingres is started at Berkeley.
1974	IBM develops a relational database called System R during 1974–1975. An improved version is developed and tested during 1978–1979. System R uses the query language SQL, which eventually becomes an industry standard.
1980	IBM releases SQL/DS, a relational database for mainframe computers.
1980's	Ingres Corporation, Britton-Lee and Sybase commercialize the technology developed by the Ingres and System R projects.
1983	ANSI/SPARC survey finds over 100 relational database implementations.
2002	The open source MySQL database has an estimated 4,000,000 active installations worldwide, with up to 27,000 downloads per day.

Table 4.6: The adoption of the relational database by businesses took several decades and generated a lot of market clutter. This table is adopted from [99] and [37].

4.10 Case Study: Adoption of Linux

This case study is about the adoption of an operating system called Linux and begins in 1991, when a Finnish student named Linus Torvalds wrote the following email message asking readers of the newsgroup comp.os.miunx what features they would like to see in a new free operating system.

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix - I'm doing
a (free) operating system (just a hobby, won't be
big and professional like gnu) for 386(486) AT
clones. This has been brewing since April, and
is starting to get ready. I'd like any feedback
on things people like/dislike in minix, as my OS
resembles it somewhat (same physical layout of
the file-system (due to practical reasons) among
other things).

I've currently ported bash(1.08) and gcc(1.40), and
things seem to work. This implies that I'll get
something practical within a few months, and I'd
like to know what features most people would want.
Any suggestions are welcome, but I won't promise
I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a
multi-threaded fs.  It is NOT protable [sic] (uses
386 task switching etc), and it probably never will
support anything other than AT-harddisks, as that's
all I have :-(.
```

Year	Version	Est. Number Users	Est. SLOC
1991	0.01	100	10,000
1992	0.96	1000	40,000
1993	0.99	20,000	100,000
1994	1.0	100,000	170,000
1995	1.2	500,000	250,000
1996	2.0	1,500,000	400,000
1997	2.1	3,500,000	800,000
1998	2.1.110	7,500,000	1,500,000
2000	RedHat 6.2	NA	1,500,000/17,000,000
2001	RedHat 7.1	NA	2,400,000/30,000,000

Table 4.7: The early growth of Linux. The estimate for the number of lines of source code for period 1991-1998 are for the entire distribution, while the estimates for 2000 and 2001 are for only the Linux kernel, as well as for the entire distribution. Sources: [86], [156], and [157].

The first release of Linux was in 1991 and within five years there were over 1,000,000 users.

This is amazing for several reasons. The first release of Linux was written by a student and later versions (to this day) are written by a community of volunteers. Linux is an example of what today is usually called open source software, since the operating system (and its source code) can be downloaded free of charge. Linux was able to reach 1,000,000 end users and Main Street in less than five years despite some significant disadvantages: 1) it competed against operating systems developed and maintained by professional engineers working at Microsoft and Sun Microsystems; 2) it had essentially no marketing budget and could not hire expensive consultants to help it reach Main Street; and 3) several vendors launched campaigns to discredit it and open source software in general.

The Mythical Man-Month. A good question to ask is why does the open source software movement works at all? Although most open source software projects involve just a few programmers, larger ones, such as the development of Linux, involve hundreds of programmers, all of whom are volunteering. Any large software project is quite challenging; until the success of Linux, few people would have predicted that a software project as complex as the development of an operating system could be completed by a team of volunteers.

The basic problem is a fundamental one: adding more programmers typically slows down large projects. This is because adding three times as many programmers produces three times as much code but usually creates more than three times as many interfaces in the code, and interfaces are often associated with software bugs.

Here is a simple example to explain the basic issue (in practice the issue is not quite this simple). Assume that each day, a programmer produces 500 lines of code containing 5 functions of 100 lines each. Assume that the 5 functions developed by that programmer work well together. But now, that programmer may find herself using the five functions of a second programmer, as well as five functions of a third programmer to complete certain tasks. This produces five times five times five or 125 combinations to check for errors, possibly subtle ones.

In this sense, although the amount of code increases linearly with the number of programmers, the number of interfaces, and hence places for possible bugs, grows geometrically. This is a well recognized problem and was explained carefully over twenty years ago in a class book called the Mythical Man-Month [Brooks:1995].

The term "mythical man-month" comes from the following observation. It is tempting to estimate project sizes as follows. Estimate the number of lines of code in the final project. Divide by the number of lines that each software engineer can code each month to get the number of person-months required. This almost never works since doubling

the number of software engineers can easily *increase* the amount of time required to complete the project for the reason above.

Successful open source projects share a number of characteristics, including some of the following:

- **Successful open source projects typically contain small, well designed cores.** The small core is written by just a few programmers, perhaps one, and the rest of the system, perhaps millions of lines, interfaces to it with simple, well-defined interfaces. Of course, this approach is often also used by companies that develop software.

- **Open source software is often written by professional programmers.** Although open source software can be obtained without charge, open source software is often written by professional software engineers. This occurs for several reasons: first, professional software engineers like writing code and especially like writing code that is elegant, important, and widely used. The right open source projects give them an opportunity to do this. Second, some companies today have business models that charge for services that are based in part upon open source software. For example, a database consultant can charge for setting up and maintaining a complex database that is built using the open source database MySQL and the open source web application development system PHP. If the success of his project requires helping fix a bug in MySQL or developing a small extension to PHP, then he is often happy to help.

- **Extensive testing.** Software is buggy. There is no way around this. The benefit of open source development is that there are large number of software engineers available to test and debug the code on a wide variety of systems in a wide variety of configurations. Bugs get identified quickly and fixed quickly.