1
Invisible Engines

But what . . . is it good for?
—*Anonymous engineer at the Advanced Computing Systems Division of IBM, 1968, commenting on the microchip.*[1]

INSIDE THIS CHAPTER

- The definition and history of software platforms
- The businesses powered by software platforms
- The basic economics of software platforms
- The plan of the book

Many modern products run on software platforms. Car engines and navigation systems have one, as do jumbo jets and the handheld devices we use for emailing and organizing ourselves. Video game consoles from Atari to Xbox are based on them. French debit cards have included them for years; these "smart cards" may eventually replace the magnetic stripe cards that are standard in the United States. Sophisticated mobile telephone services such as i-mode in Japan are based on software platforms. Personal music devices are as well. And, of course, all sorts of business and home computers also have them.

Software platforms are based on computer code. The code tells the microprocessors and other hardware components what to do. It is what

1. Caryn Yacowitz, Vittorio Zaccaria, Mariagiovanna Sami, Cristina Silvano, and Donatella Sciuto, *Power Estimation and Optimization Methodologies for Vliw-Based Embedded Systems* (Norwell, Mass.: Kluwer Academic Publishers, 2003).

makes your computer do calculations, or your personal music device play songs. And it provides services to applications, such as accessing the hardware or providing features that many applications would otherwise have to include themselves. It is what makes handwriting recognition possible on personal digital devices and enables your employer's human resources software to work on the company's computer system.

Yet these remarkable software engines are invisible to most of us. Their creators write them in a language that looks almost human. They then use other code to translate what they have written into machine language—combinations of 0s and 1s that microprocessors understand. Those digital data are then transferred to the physical memory or storage in the device itself.

Some software platforms are famous. Linux, the Mac OS, and Windows are household names. You cannot really see or touch these products, but at least you can buy a CD and a hefty manual. Others are known to many business users: z/OS, Solaris, and Unix, for instance. Many are known only to a few, such as Symbian for mobile phones or GeoWorks for handheld devices. Others, including the software platforms that are the real brains behind devices such as the Sony PlayStation or Tivo's digital video recorder, are truly anonymous.

Software platforms have generated great wealth. Windows has provided about 40 percent of Microsoft's revenues in the last decade.[2] It has helped make Bill Gates the richest man in the world. Linus Torvalds has become a modern icon as a result of writing the first version of the famous open-source platform, Linux. And software platforms have been partners in some of the most successful technological marriages of the last quarter century: the Macintosh, iPod, PalmPilot, Sony PlayStation, and Xbox are among the better known hardware-software platform couples.

The computer revolution has been changing our lives now for fifty years, at an accelerating rate, and much has been written about it. Many of the companies, products, and entrepreneurs behind this revolution have become household names. Stories of Steve Jobs and Steve Wozniak building the first Apple computer in their garage and Bill Gates getting

2. Microsoft 10-Ks, available from sec.gov.

the best of IBM are almost folklore at this point. Economists have written a fair amount about the computer industry, and business writers have scoured its history in search of the drivers of great success.

Yet little has been written about software platforms. This is not necessarily remarkable. They are not well-defined products like toothpaste. The software platform used in i-mode does not compete directly with the software platform used in Web servers. And they are not components, like the engines sold to automobile companies or even the chips sold to computer device manufacturers. There is no software platform industry defined in government statistics. Rather, software platforms are a technology—though one based on a written language—that can be deployed in a vast range of industries for a great multitude of purposes.

Many economic threads, however, tie diverse software platforms together. The most critical of these ties is their potential for supporting a multisided business—one in which value is created by bringing together on the same platform multiple distinct groups of customers who need each other in some way. Businesses that cater to the singles scene are one example of this sort of business. Heterosexual nightclubs must get men and women together in the same place. Shopping malls are also multisided: their developers create platforms that attract both merchants and consumers. Similarly, many software platforms provide services to application developers and platform users. Like shopping malls, they also provide a common meeting ground where one side can sell to the other side.

Once the multisided potential of software platforms is recognized, other similarities among businesses based on them become apparent, as do some intriguing differences.

Many charge one customer group little or nothing for using the platform. If you want to write applications for the Symbian operating system that runs on mobile phones, you can get all the necessary tools and information for very little money. The same is true for Apple, Microsoft, Palm, and most other software platform vendors. They make their money mainly from users. Manufacturers of video game consoles also have a skewed pricing model, but they make their money mainly from developers. Consumers can buy Sony PlayStations, Xboxes, and other consoles for prices that sometimes do not even cover manufacturing costs.

Manufacturers make their money mainly from game developers, who pay royalties to gain access to the information required to write games for these consoles.

Most successful software platforms have exploited positive feedbacks (or network effects) between applications and users: more applications attract more users, and more users attract more applications. Nurturing both sides of the market helped Microsoft garner thousands of applications and hundreds of millions of users for its Windows platform. The same strategy worked for Sony PlayStation in games and Palm in personal digital devices. But some software platform vendors have invested little in providing services to application developers. That was true for IBM's mainframe operating system for many years, and is still true for many manufacturers that make software platforms for dedicated devices such as ATM machines.

Software and hardware platforms have a symbiotic relationship. Neither could perform without the other. Businesses have adopted various ways of dealing with this relationship. Some have tightly integrated their hardware and software platforms; video game console companies are one example. Others have focused on the software platform and treat much of the hardware side as they do applications. Microsoft more so than most has operated a three-sided platform that tries to get users, application developers, and hardware manufacturers on board.

The multisided potential of software platforms is not their only common feature. They share all the characteristics of complex software. They are designed, written, and debugged almost entirely by humans. Much of this work is drudgery, but some of it requires solving difficult puzzles and writing sophisticated mathematical algorithms. Once created, a software program is cheaper to replicate and distribute than a book. After it sells enough copies to cover the costs of creating it, it becomes a money machine: each copy generates revenue at little extra cost. But, as with books, recorded songs, movies, and other information goods, this revenue is at risk from pirates, people who make copies for free. The intellectual effort that went into the creation of the program is also at risk. Most software businesses distribute their code only in almost indecipherable machine language and secure legal protections such as copyrights and patents to deter theft of their intellectual property.

Some people object to selling software platforms (and other software), and especially to keeping their code secret. They believe in what is known around the world as *software libre*. In some ways, they long for the earlier days of computing. Computer companies such as IBM used to include software with their machines; buyers did not consider it something they paid for separately. For many years after the birth of modern computing, software was shared among colleagues. The notion of selling software, and especially software platforms, did not arise until the 1970s, almost a quarter century after the sale of the first commercial computer. The software industry started booming in the 1980s and has generated over $500 billion in sales worldwide in the last three years (we leave all currency figures in their original amounts and do not adjust for inflation).[3] The free software movement has tried to return to the more collegial approach of the industry's youth. Its greatest success is Linux, which is developed through a collaborative process among programmers around the world working through the Internet and coordinated through various committees. Linux is known as open source because you can read the programming code in which it was written. It is available for free, subject to some important restrictions we discuss later.

Most software platforms share another feature: they grow over time. Version 9.0 of the Red Hat Linux OS, introduced in 2003, has 50 million lines of code, compared with 9 million for Version 5.0, which came out in 1997. The same is true for the Mac OS. It started with one-fifth of a megabyte in 1984 and takes up more than a thousand megabytes today.[4] Software platforms grow because they do more things—they provide more features for application writers, end users, or both. In some cases, they are just taking advantage of faster microprocessors and larger memory. In others they are absorbing features that were once performed by separate applications. As more people began wanting to connect to

3. Richard V. Heiman, Sally Hudson, Henry D. Morris, Albert Pang, and Anthony C. Picardi, "Worldwide Software Forecast Summary, 2003–2007" (IDC report no. 30099), September 2003; Richard V. Heiman and Anthony C. Picardi, "Worldwide Software 2004–2008 Research Summary" (IDC report no. 31785), August 2004.

4. http://applemuseum.bott.org/sections/os.html; http://www.apple.com/macosx/techspecs/.

the Internet, for example, software platforms started including communication protocols that made that easier to do.

The following pages document patterns and anomalies across businesses based in whole or in part on software platforms. These regularities and irregularities are the source of insights that we hope will be useful to entrepreneurs and investors as well as economists. The patterns result from the underlying economics of software platforms. The skewed pricing structures that appear for most software platforms are common in other multisided platform businesses. Not surprisingly, the anomalies are both more intriguing and harder to rationalize. Economics, however, can narrow down the possible explanations. The differences between software platforms for video game consoles and PCs could result from path dependence (they started from different points, which determined their futures) or fundamental differences in economics (the manufacturer needs to make a market for game consoles to induce application developers to write, or consumer tastes for applications differ from those for games).

The challenges faced by software platform businesses are encountered by many other businesses that are multisided, or could be. Deciding whether and when to rely on outsiders for crucial complementary products is critical. Microsoft has built a software platform empire through partnering with many other firms that produce complements for it. But Apple's iPod/iTunes music platform has found success by doing everything from making the music device, designing the software, and running the music store. Pricing is key as well: finding the right balance between the various sides is one of the hardest problems faced by platform businesses. 3DO's innovative game platform died a quick death when it priced its consoles too high and its royalties for game developers too low. An ill-chosen pricing strategy put Microsoft's Xbox on the brink of disaster but one that it averted in time. Other platform businesses could learn from how software platforms load features to get and keep both sides on board. Whether you use Windows, Linux, or the Mac OS, most of the code on your hard disk has no direct value to you. Much of it is there for developers of applications, most of which you will never use. Other portions are there to provide esoteric features that only a few of us use.

Making Computers Smaller

Products based on software platforms abound because of the microprocessor revolution that began in the 1970s.

The first general-purpose electronic computer, ENIAC, was created during World War II for calculations that helped aim artillery toward targets. Based on 18,000 vacuum tubes, it was 100 feet long, 8.5 feet high, and several feet wide. The development of the transistor, which began in the late 1940s, led to the second generation of computers. The transistor serves the same function as the vacuum tube but is much smaller, requires much less power, and is much more reliable. Second-generation computers were thus smaller and less expensive to run. Third-generation computers were made possible with the invention of the integrated circuit in 1959.

The integrated circuit, which combined several transistors and other circuit elements into a single component, not only further reduced the size and price of computers but also made them faster. Admiral Grace Hopper, a pioneering software programmer, was famous for carrying around a "nanosecond"—a footlong piece of telephone wire representing the maximum distance electricity can travel in one nanosecond. She used it to illustrate that computers had to be small to be fast. And computers did get smaller. The popular IBM System/360 Model 30, introduced in 1964, took up about 106 cubic feet.[5] Minicomputers were even smaller. Digital Equipment Corporation's PDP-8, introduced in 1965, was only about 8 cubic feet. Minicomputers were small enough that manufacturers could, for the first time, integrate computing power into laboratory devices and other equipment.

The current generation of computers began with the development of a microprocessor at Intel. The microprocessor packs the whole central processing unit (CPU), which is often called the brains of the computer and which involves many transistors, onto a single semiconductor chip. This has made it possible to provide massive amounts of computing power in small devices. Produced in 1971, the Intel 4004 was the first microprocessor. It had 2,300 transistors on a silicon wafer the size of a

5. http://homepages.kcbbs.gen.nz/nbree/saga.html.

ladybug and could perform 60,000 instructions a second. Three years later Intel introduced the Intel 8080, which had 4,500 transistors on a silicon wafer of about the same size as the 4040 yet could perform more than 500,000 instructions per second. It is considered the first general-purpose microprocessor, and its release marked the birth of the microprocessor industry. It soon spawned the first microcomputer, the Altair 8800, and the first video game system, Midway's Gun Fight arcade game.

The computing power of microprocessors depends on the number of transistors on the chip. Manufacturers have approximately doubled that number about every 18 months since the 1970s (this regularity is known as Moore's Law).[6] A computer science textbook published in 2002 notes that "the highest-performance microprocessors of today outperform the supercomputer of less than 10 years ago."[7] That microprocessor is the size of a fingernail; the supercomputer filled a room.

The price of computing power has declined as well, in part because microprocessor production allows for extensive scale economies. This decline has been dramatic. For example, the number of integer operations per second per dollar grew more than 500-fold between 1990 and 2004.

Other hardware advances have helped miniaturize computing devices. The most notable is the decrease in the size and cost of disk storage. During this same time period, the amount of magnetic disk storage that could be purchased with a fixed dollar budget increased by about 500 times, and the disk density or the number of megabytes per square inch of disk surface increased by more than 1,200 times.

Advances in technology and computer design have provided ever smaller and cheaper computers. A comparison of specifications makes clear that a typical $1,000 computer bought around 2003 had greater computational performance, main memory, and disk storage than a $1,000,000 computer bought around 1980. Even more remarkable are consumer products that are based on computing devices.

6. Paul Freiberger and Michael Swaine, *Fire in the Valley*, 2nd ed. (New York: McGraw-Hill, 2000), p. 377.

7. John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach,* 3rd ed. (New York: Elsevier, 2002), chap. 1.

At less than 4 cubic inches, the 2004 iPod mini can easily fit in a shirt pocket. It has two 80-MHz microprocessors and can store more than 1,000 pop songs on its 6-gigabyte storage disk. At $249, it is several times more powerful than the multi-million-dollar IBM System/370 available in 1970.

The Growing Family of Computer-Based Products

The microcomputer industry grew rapidly as a result of these favorable technological and cost trends. In 1981, shortly after IBM added its microcomputer to the ones already introduced by Apple and others in the late 1970s, 344,000 microcomputers were sold in the United States. By 2004 there were an estimated 822 million computers in use worldwide. Almost every office worker in the United States now has one, and 56 percent of American households have at least one.[8]

The video game device was the first mass-produced good based on the microprocessor that was not a traditional computer. The early devices, introduced in the late 1970s, played a single game. Over time, video game consoles were developed that rivaled the most sophisticated personal computers. These were able to play numerous games that were compatible with their software and hardware. By 2002 the video game industry had reached $21 billion of annual revenue from the sale of consoles and games, surpassing the movie industry's $19 billion in annual box office revenues that same year.[9]

The increasing power and decreasing size of microprocessors and other hardware components made handheld computers feasible by the late 1980s. The Apple Newton was the first of these. The original Newton was about the size of a VHS cassette and functioned basically as an electronic notepad. It was technically interesting but a commercial failure. A few years later Palm introduced the PalmPilot, which had widespread appeal and helped create the handheld industry. At first these products were used mainly as sophisticated organizers; they competed with

8. Rex Crum, "*Computer Industry Almanac* Sees 1 Billion PCs by 2007," *CBS Market Watch*, March 9, 2005 (available from Lexis-Nexis); http://www.census.gov/population/socdemo/computer/ppl-175/tab01A.pdf.

9. "Gaming's New Frontier," *The Economist*, October 2, 2003.

Filofax. Over time they added Internet browsing and wireless email. In 2004, more than 31 million handheld devices ranging from BlackBerries to Treos were sold worldwide.[10]

Many, if not most, new mobile phones have calculators, games, and other computer-based features, and it is often possible to access more applications by downloading them directly from the Internet or downloading them to a PC and transferring them to a mobile phone. As wireless networks have gotten more sophisticated, wireless telephone companies have turned their phones into Web portals through which users can obtain various kinds of content and send email. Japan's DoCoMo was the pioneer here. Vodafone and other mobile networks have followed. There were more than 1.5 billion mobile phones in use worldwide in 2004.[11]

Digital music devices started becoming popular in the early 2000s. Their roots go back to the PC. Starting in the early 1990s, PCs could play CDs, and by the mid-1990s they could store and retrieve digital music tracks on disks. Various formats were developed for transmitting digital music over the Internet, including MP3, and several "media players" became popular for playing and manipulating music on PCs. Stand-alone MP3 players were introduced in the late 1990s. The industry is now synonymous with the iPod, introduced in 2001. More than 32 million handheld music devices were sold in 2003, and this industry is expected to expand dramatically with the increasing popularity of downloading music.[12]

Microprocessor-based computing devices were incorporated into many other products starting in the 1980s. ATMs are one example. Intel 8086 microprocessors powered cash dispensers in the late 1970s. Over time, ATMs have become PC-compatible devices that use stripped-down versions of common PC software platforms such as Windows or Linux. Cars are another example. Indeed, software gremlins are

10. David Linsalata, Kevin Burden, Ramon T. Llamas, and Randy Giusto, "Worldwide Smart Handheld Device 2005–2009 Forecast and Analysis: Passing the Torch" (IDC report no. 33415), May 2005, table 1.

11. http://www.itfacts.biz/index.php?id=P2193.

12. Susan Hevorhian, "Worldwide Compressed Audio Player 2004–2008 Forecast: MP3 Reaches Far and Wide" (IDC report no. 31811), August 2004, table 4.

behind a spate of complaints about windows going down on their own and temperature control systems turning up the heat on hot summer days.

The French payment card system started incorporating microprocessors into their debit cards in the late 1980s.[13] These were used mainly for verification. Cardholders entered their personal identification number on a reader that verified it against the number contained in the chip. These "smart cards" have gotten more capable and cheaper to produce. The major card systems have worked on developing software and hardware standards for these cards. Smart cards are being used at colleges to keep track of meals, for social welfare programs, and for secure purchasing over the Internet. Several card issuers in the United States have introduced "contactless" chip cards that are simply waved at a device at the point of sale. It is likely that within a decade, most of the cards used for payment around the world will be smart cards and thus based on small computers. (In 2003, there were already 220 million smart cards involved in banking-related uses alone worldwide.[14])

Software Platform Elements

A complete software platform does everything from telling the microprocessor to turn switches on or off to providing a host of full-fledged software features for application developers that save them the time of writing those features themselves. Many software platforms, though, are based on different software programs that provide different portions of these services. The boundaries between these programs are not always clear in practice and can change over time. To add to the confusion, these programs have names that are sometimes used interchangeably even though the programs do somewhat different things.

Moving from controlling the microprocessor to serving application developers, it is useful to distinguish four kinds of platform-related programs.

13. David Evans and Richard Schmalensee, *Paying with Plastic*, 2nd ed. (Cambridge, Mass.: MIT Press, 2005), p. 302.

14. http://www.epaynews.com/statistics/scardstats.html#7.

Software platforms are often, but not always, *operating systems*. The nucleus of a computer operating system is generally called the *kernel*. It manages the processors, memory, input and output, and certain support functions. It controls the hardware to calculate 2 + 2 and sends "4" to an output device. This is the first thing Linus Torvalds wrote to get Linux going.

The operating system generally also assists application programs in other ways. For example, it may help programs display complex graphics, such as a three-dimensional graphical depiction of 2 + 2 = 4, on a mobile phone screen or computer monitor. PC operating systems such as the Mac OS X are usually full-fledged software platforms; only the operating system stands between the hardware and the applications. In this case the terms *operating system* and *software platform* are synonymous.

Middleware typically refers to software that specializes in providing services to application developers. It does not have a kernel, and it relies on the operating system to control the hardware. Some middleware sits on top of an operating system kernel and does everything besides basic hardware support. For example, the operating system for the Sony PlayStation is little more than a kernel; game developers such as Electronic Arts write their own middleware, which provides support for various PlayStation games they create. Other middleware leaves tasks to an operating system that go beyond those generally performed by the kernel. And some middleware sits on top of a software platform: it may help applications run on many different software platforms, and it may compete with the software platform for the attention of application developers. That is the case with Sun's Java technologies.

Another critical aspect of a software platform's architecture is whether it is *open* or *closed*. With an open platform, anyone with the right technical knowledge can obtain access to the services provided by the platform or its underlying elements. Most PC platforms are open: one can write applications for the Mac OS or Windows without getting permission from the manufacturers. With a closed platform, only those with permission to use the platform can benefit from its services. Most game and mobile telephone platforms are walled off. Programmers have to get a "certificate" to get access to the operating system and hardware

on these computing devices. Hacking is still possible, but much more difficult.

The Plan of the Book

This book is organized into four major parts. The next two chapters provide background. Chapter 2 describes software platforms from a technological standpoint. What do they do? How do they do it? How are they created? Chapter 3 considers their key economic aspects. It introduces the economics of two-sided markets, which is critical for understanding the nature of the demand for software platforms. It also explores characteristics that software platforms share with other software products and many information goods.

In the second part, Chapters 4 through 8 analyze important industries in which software platforms have played a prominent role, either as a standalone product or as an important component in a computer system consisting of hardware and software. The chapters in this section examine, in order, PCs, video games, handheld devices, mobile telephones, and digital music players and devices. These chapters are not intended to provide a complete history of these industries. After some initial background discussion, each chapter focuses on the strategies that companies pursued in these industries over time. In particular, we look closely at efforts to get multiple customer groups on board the platform, pricing, product design, and the organization of the supply chain and ecosystem. To help the reader understand the evolution of these industries, we present a timeline of the events we focus on for each chapter. (A historian of these industries would no doubt create a somewhat different timeline of key events.)

The third part, Chapters 9 through 11, examines the similarities and differences across these industries on several critical dimensions. Chapter 9 compares and contrasts business strategies for supplying software platforms. The focus is on business integration among the various levels of providing computer systems. Why have some companies chosen to provide both the hardware and the software platform, while others in the same segment have specialized in the software platform? How does the decision to disintegrate vertically vary over time, and why? Chapter

10 looks at pricing. We show that all these industries charge one customer side a "low" price. With the exception of video game consoles, all provide software developers with inexpensive access to valuable features in the platforms. What is the reason for the single exception? Chapter 11 examines the bundling of features. Like most information goods, software platforms combine features that attract very different groups of customers. What is extraordinary is the extent to which software platforms grow through the accumulation of features.

The fourth part, which consists of just Chapter 12, focuses on the role of software platforms in the process of creative destruction. Many software platforms have marched from the narrow market in which they were first introduced into other markets. PCs and video game consoles are both trying hard to get into your living room. Mobile phones are moving into digital cameras, personal music devices, personal digital assistants, and payment cards. The boundaries between software platforms and the industries they power are blurring. Many call this convergence. But it is also a life-and-death struggle for the businesses involved. As the iconic Pilot goes the way of the typewriter, the Palm OS may evolve into a successful mobile phone software platform, or it may just wither away. As some software platforms—and whole product categories—die, others are born.

Chapter 12 also looks at the role of two types of software platforms that were born shortly after the start of the commercial Internet in 1995. Both are Web-centric platforms. Both are designed to facilitate transactions between buyers and sellers in the economy. The code sits on servers that are connected to the Internet. One type is based on conducting auctions, the other on conducting searches. As of 2006, eBay and Google are the leaders in these respective categories.

Although these companies are not in the business mainly of providing software services to users or developers, they have opened their software code to developers and are providing services that facilitate developers writing applications. These Web-centric platforms will be sustaining a vast economy of developers, based on the experience of other software platforms we have examined. The symbiotic relationship between and among the platforms, developers, buyers, and sellers is expected to lead to profound changes in the retail economy.

INSIGHTS

- Software platforms are invisible engines based on written computer code. Software platforms power, to varying degrees, many modern industries, including digital music, mobile phones, on-line auctions, personal computers, video games, Web-based advertising, and online searches.
- Starting in 1970, the microprocessor revolution has stimulated the development of software platforms for diverse computing devices, enabled software platforms to migrate to smaller devices, and helped software platforms do more over time everywhere they are used.
- Software platforms usually provide valuable services to people who use computing devices, developers who write applications, and makers of computing hardware.
- Most businesses based on software platforms follow *multisided* strategies to get users, developers, and hardware makers on their platforms. These strategies are critical for harnessing positive feedbacks. For example, users value more applications, and applications developers value more users.

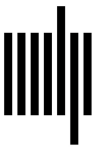
Invisible Engines

How Software Platforms Drive Innovation and Transform Industries

By: David S. Evans, Andrei Hagiu, Richard Schmalensee

Citation:

Invisible Engines: How Software Platforms Drive Innovation and Transform Industries
By: David S. Evans, Andrei Hagiu, Richard Schmalensee
DOI: 10.7551/mitpress/3959.001.0001
ISBN (electronic): 9780262272421
Publisher: The MIT Press
Published: 2008



The MIT Press

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu or write to Special Sales Department, The MIT Press, 55 Hayward Street, Cambridge, MA 02142.

This book was set in Sabon by SNP Best-set Typesetter Ltd., Hong Kong.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Evans, David S. (David Sparks)
Invisible engines : how software platforms drive innovation and transform industries / David S. Evans, Andrei Hagiu, and Richard Schmalensee.
p. cm.
Includes bibliographical references and index.
ISBN 0-262-05085-4 (alk. paper)
1. Application program interfaces (Computer software). 2. Industries—Data processing. I. Hagiu, Andrei. II. Schmalensee, Richard. III. Title.
QA76.76.A63 E93 2006
005.3—dc22

2006046629

10 9 8 7 6 5 4 3 2 1