This book will help you negotiate, draft, and understand information technology contracts. Specifically, it will help you with software licenses and other software transfers, cloud computing agreements, and technology professional services agreements. It addresses contracts between businesses, as well as business-to-consumer and business-to-government contracts. It also addresses both offline contracts and contracts related to the Internet and e-commerce.

This book is for both lawyers and nonlawyers. The text stays away from technical jargon—“legalese,” “engineerese,” and “programmerese”—and where it absolutely can’t avoid jargon, it provides a definition. In other words, this book is written in simple English, like a good contract.

You can use this book as a training manual or a reference guide or both. If you’re training, read this book cover to cover. It provides an overview of the key technology contracting concepts.

If you’re after a reference guide, you can pick and choose the chapters to read. When you’re negotiating a contract, or reading or writing one, look up the various clauses to learn what they mean and what’s at stake. You’ll find sample language in each chapter, which you can incorporate into your own contracts. Plus, if you visit this book’s website, http://TechContracts.com, you can copy the longer sample clauses and paste them into your document. You’ll also find several full-length contracts at the website, which you can download and revise to fit your deals.

Finally, you can also use this book’s table of contents as an issue spotter—as a checklist of clauses to consider.

This book can’t replace a lawyer—or a colleague with more information technology (IT) experience, if you are a lawyer. But it can help you understand your lawyer or colleague. And whether you have
legal help or not, the better you understand your contracts, the more
effective you’ll be.

I’m a technology lawyer, and this book grew out of seminars I
 teach, for both attorneys and nonattorneys. At the end of the program,
students often asked where they could learn more—if I knew a good
book on IT contracts. Most of the books I knew were massive tomes on
intellectual property or contract law. They’re written for lawyers only,
and their more practical lessons are spread across hundreds or thou-
sands of pages. I’ve learned much of my trade on the job, rather than
from a book. I’ve served as a technology lawyer with a global firm, as
general counsel for a publicly traded software company, and as vice
president of business development for an Internet start-up. I now prac-
tice through my own technology-focused law firm in San Francisco
and the Silicon Valley. The material for my seminar came from the
contracts I’ve negotiated and written in those positions. I’d never seen
a really user-friendly outline of the issues. So I wrote this book.

The rest of this introduction provides more detail about the types of
contracts this book covers. It also explains the structure of a contract
and of this book and offers a few explanations that will help you get
the most out of your reading. Finally, it provides a short explanation
of some IT industry language—just a little, particularly regarding
cloud computing—and then offers three lessons about contracting
in general.

**Subject Matter: Types of IT Agreements**

This book addresses four principal types of IT contracts:

1. **Software License Agreements** and **Software Ownership
   Agreements** transfer intellectual property rights from the
   vendor to the customer. They include end user licenses,
   enterprise licenses, distribution contracts, assignments, and
   work-for-hire agreements. In all these deals, the customer
   gains, at a minimum, the right to put one or more copies
   of the software on its computers: the right to make copies.
   These contracts need an intellectual property (IP) transfer
   because the right to make copies is protected by copyright
and other IP laws.¹ The rights transferred might be limited, like the right to make a few copies or to distribute the software: a software license. Or the customer might receive all IP rights and become the software’s new owner: a software ownership transfer.

2. **IT Professional Services Agreements** call on the vendor’s staff to help the customer. The vendor’s professionals are going to do something, rather than simply making software or other technology available (as in a software license or cloud services agreement). IT professional services include system integration, tech support, website development, software maintenance, and technology consulting.

3. **Cloud Services Agreements** call on the vendor to host a cloud computing system and to give the customer remote access, usually via the Internet. They’re sometimes called Cloud Hosting Agreements or Cloud Computing Services Agreements. A cloud services agreement is neither a professional services contract nor a software license. It’s not professional services because computers play the key role in providing the service, not human professionals. And it’s not a license because the customer gets no copies of the cloud computing software—just remote access. (For more on cloud services, see “A Little Industry Language, Particularly re Cloud Computing,” later in this Introduction.)

4. **Combination Agreements** call on the vendor to provide some combination of software, professional services, and cloud services. Computer programming contracts, for instance, usually call on the vendor to write software, a professional service, and to transfer IP rights in that software, a software license or ownership transfer. Cloud hosting and support contracts call on the vendor to host software (or other technology) and make it available to the customer, a cloud services offering, and to help the customer figure out how to use that software, a professional service. A combination contract works like two or more contracts in one. It needs terms appropriate for two or three of our contract types: software licensing, professional services, and/or cloud services.

---

¹ For an explanation of IP and its role in tech contracts, see Appendix 1 (“Intellectual Property”).
This book will also help you with purchase and lease agreements for computers and other IT hardware. That’s because many of the clauses discussed here appear in hardware contracts too, and some of the contracts listed above can include hardware purchases. A software customer, for instance, might license software and buy computers to run it, all in one contract. That said, hardware agreements involve some terms you don’t see in other types of IT contracts, like leases, security interests, and shipping terms. This book doesn’t cover those clauses.

Finally, this book addresses both private contracts—business-to-business and business-to-consumer—and government contracts. A government contract is an agreement between a private vendor and a government agency customer. It can involve any of the contract types described above. Most government contracts use language required by government contracting laws and rules, like the federal acquisition regulations (FARs) and state procurement regulations. This book doesn’t address those rules, but it should still help you understand the language they require. And of course, where the government doesn’t require its own language, you can use this book’s sample clauses.

The Structure of a Contract and of This Book

IT contract terms can be organized into three groups: prime clauses, general clauses, and boilerplate clauses. This book is organized the same way.

The prime clauses express the deal’s central terms. There, the vendor grants a license or other rights to software, or promises to provide professional services or cloud services—or some combination of the three, in a combination contract. The customer, on the other hand, promises to pay. This book addresses prime clauses in Part I.

---

2. Many government contracting rules require terms unique to government procurement, like “buy American” provisions and clauses on conflicts of interest or use of recycled paper. This book doesn’t address those terms.

3. You might not find these terms outside this book (the same goes for “combination contract,” above), but they’re meant to express common views of these clause types. The first edition of this book used “transactional clauses” instead of “prime clauses,” and “supporting clauses” instead of “boilerplate clauses.”
The general clauses account for most of the contract. They cover everything not addressed in the prime clauses or the boilerplate clauses, and they’re usually the most heavily negotiated. This book addresses general clauses in Part II.

Boilerplate clauses cover the theoretically noncontroversial mechanics of a deal: terms on independent contractor status, contract interpretation, choice of law, etc. IT professionals tend to put most of these terms at the end of a contract. This book addresses boilerplate clauses in Part III.

Contracts usually start with two sets of boilerplate clauses: the introduction (including “recitals”) and the definitions. From there on, you should organize your clauses the way they’re listed above: prime clauses, then general clauses, then the remaining boilerplate clauses. That makes agreements easy to understand. Unfortunately, though, you’ll probably run across contracts with these clause types jumbled together in no particular order.

Using This Book

The following four brief notes and explanations will help you get the most out of this book.

First, almost any contract you write should be customized to fit your particular deal. So if you insert one of this book’s sample clauses—which is what they’re here for—don’t do it thoughtlessly. You may need to edit it. Think through the unique issues raised by your deal. “We know the Windows version of our software has some serious bugs, so we don’t want to give a broad warranty.” “We’ve got a small IT department, so we need extra support.” “Our CFO gets hives if we give personal injury indemnities.” This book offers building blocks for a contract addressing issues like those, but the customizations are up to you.

Second, this book addresses U.S. law. That doesn’t mean it’s useless for contracts under other countries’ laws. Most contract clauses mean what they say, regardless of the underlying legal system. But with some clauses, the underlying law really will affect the meaning. So if you’re working outside the U.S., consider help from a local lawyer.

4. See Chapters III.A ("Introduction and Recitals") and III.B ("Definitions").
Introduction

Within the U.S., the 50 states have similar contract laws, and federal law governs some of the issues discussed here, particularly related to copyrights and patents. So state law variations won’t often lead your IT contracts astray. But sometimes state law variations really do matter. That’s another reason to consider help from an experienced lawyer.

Third, like most contracts, the examples in this book use defined terms. When a contract creates a concept and uses it more than once, it usually defines it. For instance, a contract might list the vendor’s services in Section 2, then mention them over and over in other sections. Rather than listing the services repeatedly, the contract defines the list as the “Services.” Whenever the contract refers to the “Services” with a capital S, it means the whole list. This book’s sample clauses work the same way: capitalized words that aren’t proper names represent defined terms—e.g., “Software,” “Effective Date,” “Statement of Work,” this “Agreement.” (Some contracts mark defined terms with all caps instead—e.g., the “SERVICES.”) The same goes for sets of initials in all caps, like “NDA” (for nondisclosure agreement). In this book, the sample clause often won’t supply the definition. That’s because, in a real contract, another section would define that term. Obviously, in your contracts, you should provide the definitions somewhere.

Fourth, most of this book’s sample clauses use the defined terms “Vendor” and “Customer,” and so does the text. Contracts you’ve worked with may use other names. “Vendor” stands in for “Licensor,” “Provider,” “Transferor,” “Assignor,” “Seller,” and “Consultant,” among others. And “Customer” stands in for “Licensee,” “Transferee,” “Assignee,” “Recipient,” “Buyer,” and “Client.” This book favors “Vendor” and “Customer” because they’re generic. But the text and sample clauses do occasionally use names like “Distributor” where necessary to avoid confusion.

A Little Industry Language, Particularly Re Cloud Computing

This book includes the world’s shortest glossary, on page 271. It explains five terms you’ll see in the text: calendar (as in “calendar

---

5. This book’s first edition used “Recipient” instead of “Customer” and “Provider” instead of “Vendor.”
This book also uses some terms related to cloud computing, and they're important enough to explain up front. The key terms are “cloud computing” itself and “cloud services.” It's also worth explaining “software-as-a-service,” and while we're at it, we'll touch on two related terms: “platform-as-a-service” and “infrastructure-as-a-service.” Experts actually disagree on these terms' definitions, and many would say this book oversimplifies some complex concepts. But the definitions given here work for our purposes.

“Cloud computing” is a model for delivering software and other IT resources through a particular type of computer network. The software sits on one or more central computers—servers—and the customer's users access it remotely, from their own computers (often desktops and other client computers). Usually, access is via the Internet. The customer might host the server computers and software itself, but often the vendor (or its reseller) hosts.

Where the vendor hosts the server computers and software, this book and many IT professionals call the arrangement “cloud services” (or sometimes “hosting services” or “cloud computing services”). As you'll see in Chapter I.E (“Subscription for Cloud Services”), cloud services don't involve a software license, since the vendor keeps the software to itself; it doesn't give the customer any copies. What the vendor really provides is a service: remote access to and use of its server computers and the software. On the other hand, if the customer hosts the software, it does need a license, since it has to make copies of the software. So an agreement letting the customer host is a software license agreement. This book doesn't talk much about “customer-hosted cloud computing,” but that's not because it's unimportant. Rather, this book lumps it in with other software licenses.\(^6\)

You've probably heard a lot about a particular type of cloud services: “software-as-a-service” (SaaS). SaaS refers to a cloud service where the remotely hosted IT resource is a software application, or several applications. An application is software for users, like a word processing or contacts management program—something a human being actually sits down and uses. It's contrasted with platform

---

6. Of course, the vendor could host some cloud systems while the vendor hosts others. That would create a combination contract, with two prime clauses: a subscription for cloud services and a software license. See “Subject Matter: Types of IT Contracts,” earlier in this Introduction.
software (aka system software): a program that runs a computer, like an operating system. So in a SaaS relationship, the vendor puts software for users on its server computers and makes it available to the customer, usually via the Internet.

IT professionals talk about two other types of cloud services. In a “platform-as-a-service” (PaaS) offering, the vendor hosts a software platform. The customer installs or creates applications on that platform and uses them. And in “infrastructure-as-a-service” (IaaS), the vendor hosts server computers and other hardware infrastructure. The customer installs both platform and application software on that infrastructure, and uses them. In both cases, the customer usually accesses the systems via the Internet. This book rarely addresses PaaS or IaaS separately; it just discusses cloud services.

Three Lessons about Contracting

1. Good Fences Make Good Neighbors

Why do we sign contracts? It’s not because we want to win a lawsuit later. It’s not because we don’t trust each other. It’s not even because we’re afraid lawyers will stir up trouble if they’re not kept busy.

We sign contracts because good fences make good neighbors.

The best way to avoid arguments in a business relationship is to write down the parties’ expectations ahead of time. That list becomes a boundary marker—like a fence between neighboring yards—explaining who’s responsible for what. If the parties disagree, they can look at the list for guidance.

In other words, contracts prevent disputes—at least, good ones do. They prevent lawsuits.

Even if the parties never look back at the contract once it’s signed, it has still probably played a vital role. When people put their business expectations on paper, they often find those expectations don’t match. Just the act of negotiating a written contract will uncover many mismatched expectations. The parties can address them before starting work.

---

7. Many contracts can be oral rather than written, but some can’t. And oral contracts lack the advantages of clarity and detail.
Yes, it’s true that we sometimes fight over contracts in lawsuits. And yes, in interpreting a contract, we often talk about what a judge would say it means. But that’s only because courts have the ultimate say if the parties can’t agree. Job number one for the contract is to keep the parties out of court.

2. There Is No Such Thing as “Legalese” or “Technicalese”

You may feel uncomfortable with contracts because of the unfamiliar language they use. Don’t be intimidated. You can understand most contracts.

There really is no such thing as legalese. American contracts are written in English (or Spanish or Vietnamese or whatever language the parties speak). But contracts do sometimes use special shorthand: terms lawyers have developed to save time. And some IT contracts use “technology shorthand.” Finally, contracts sometimes use formal, stilted language with long run-on sentences. Don’t let shorthand or stilted language bother you.

If you run into an unfamiliar term in a contract—unfamiliar shorthand—don’t worry. Look it up. If it’s legal shorthand, you can probably find the definition in a standard dictionary, or online, or in *Black’s Law Dictionary*, found in many libraries. Treat technology terms the same way. Look them up in a dictionary or technical manual or online. Or ask someone with the right expertise.

Once you understand a term, feel free to use it in your own contracts. But you should also feel free not to use it. Shorthand is optional. If you do use shorthand, be sure the contract defines each technical term. Definitions can vary for IT terms like “sandboxing” and “bot,” so the contract needs an agreed definition, unless there really can’t be any doubt. Legal terms, on the other hand, often have widely accepted definitions, so you usually don’t need to define them in the contract.

As for long sentences, just take a deep breath and read slowly. The same goes for formal language. There really is no reason to use

---

8. From the West Publishing Company.
9. **Sandboxing**: separating a computer program from other programs to limit the impact of errors and security issues. **Bot**: a software program that mimics human behavior in that it’s automated (short for robot).
Introduction

terms like “heretofore” and “lex loci.” That sort of language often appears in form contracts from the olden days, when formal writing and Latin were more popular. It does crop up in modern contracts—often because someone wants to show off a big vocabulary. Be suitably impressed. Then take out your dictionary if necessary and figure out what each sentence says. And avoid terms like that in your own writing.

3. Ask Yourself “What’s Our Best Option?” Not “What’s Fair?”

Some businesspeople and lawyers ponder and argue a lot about whether proposed contract terms would be fair. I think that’s an unhelpful view of contracts, for two reasons. First, it’s hard or even impossible to define “fair” in contract negotiations. Second, a focus on what’s fair may lead you to reject deals that make economic sense, or to accept deals that don’t. The better question is: would doing the deal under these terms be more profitable than not doing it?

What does “fair” even mean in contract negotiations? Each party has a choice about whether to do a deal, so neither owes the other any particular terms. If some company insists on terms heavily slanted in its favor, and no one ever accepts them, that company’s dumb, not unfair. Or maybe it just doesn’t care whether it does any deals. On the other hand, if enough people do accept the bad terms, does it make sense to call them unfair? Does it even make sense to call them bad terms? The fact that “the market” accepts the terms legitimates them.11

So you might say “fair” can’t be defined in contract negotiations. Or you might say “fair” means “acceptable to enough people.” Either way, the guiding principle behind contract terms is leverage: whether the proposing party can get its terms often enough to do the deals it needs.

If you do focus on what’s fair, you might walk away from deals that make economic sense. If terms you don’t like seem unfair, and you can’t get the other party to budge, you’ll probably feel too screwed

10. Heretofore: before now. Lex loci: Latin for the law of the place, usually referring to a contract’s choice-of-law clause.
11. OK, that’s not entirely true, at least so far as the law is concerned. There are a few contract terms courts won’t enforce because they’re “opposed to public policy” or “unconscionable.”
to sign the contract. But that’s a poor choice if you couldn’t get better terms from anyone else, and if accepting the terms would be more profitable than dropping the project. What if you focus on best option available, rather than fairness, and recognize that the other party wants its best option too? In that case, you’ll accept the deal if it’s the best option the market has to offer—and you’ll feel good about it. You’ll only walk away if you have better options.

A fairness focus could cut the other way too. You might make concessions because the other side’s requests sound fair—despite the fact that you’ve got options better than doing the deal under those terms. If you simply ask yourself whether you’ve got better options, and the answer’s yes, you’ll refuse the other side’s “fair” terms.

I’m not suggesting contract negotiators should be androids or Vulcans, who react only to logic. Most of us want the other side to leave the bargaining table happy. And we do respond to arguments about fairness. We just need to remember how slippery the concept is, and how ultimately the market for other options shapes the definition of “fair,” rather than some objective concept of right and wrong.