

Custom Software Development Agreement

Introduction	449
Phased Agreements Work Best	449
Who Writes the Contract?	450
Contents of Custom Software Development Agreements	450
Identification of the Parties	450
Purpose of Agreement	451
Preparation of Development Plan	452
Payment	455
Payment of Developer's Costs	456
Late Fees	457
Materials	457
Changes in Project Scope	458
Delays	459
Acceptance Testing of Software	460
Training	463
Maintenance of Software	464
Ownership of Software	465
Ownership of Background Technology	470
Source Code Access	472
Warranties	474
Intellectual Property Infringement Claims	481
Limits on Developer's Liability	484
Confidentiality	487
Term of Agreement	488
Termination of Agreement	488
Taxes	489
Developer an Independent Contractor	490

Optional Provision: Nonsolicitation of Developer’s Employees491

Disputes.....492

Attorney Fees492

General Provisions493

Signatures494

Exhibits.....494

This chapter covers contracts for the developer of custom software.

It is written primarily from the developer's point of view, but contains much useful information for custom software customers as well. There is no standard definition of "custom" software. In this chapter, the term is used to mean software—usually an entire system—created specially for a particular customer. But custom software need not be created entirely from scratch. Many of the components of the system may have been used by the developer before (and may be used again for other customers). As long as the system as a whole is unique, it fits the custom software definition.

Introduction

Contrary to conventional wisdom, a developer does not benefit by starting work immediately on the basis of a handshake or a brief letter that sets forth the contract price and a vague description of the project. Unfortunately, many development projects are handled in just this way. However, this approach is a recipe for disaster. If problems later develop—particularly if the customer had unrealistic expectations or failed to understand exactly what the developer agreed to do—the developer will have no contract to fall back on for help. That piece of paper—the development contract—is the developer's lifeline. If properly drafted, it will help prevent disputes by making it clear exactly what's been agreed to. If problems develop, the agreement will provide ways to

solve them. If the parties end up in court, the agreement will establish their legal duties to each other.

Negotiating a contract can take some time and effort, but by using the phased approach advocated below, the delay should be minimized. Under this approach, the contract can be signed on the basis of a simple functional specification, without going to the time and effort of creating final detailed specifications or a delivery and payment schedule—usually the most time-consuming parts of a development agreement. These are created later as part of the first phase of the project.

Phased Agreements Work Best

It's almost always advisable to break down a custom software development project into discrete parts or stages, often called phases or "milestones." At the end of each stage, the developer delivers an acceptable product. Having the customer sign off on each phase of the project avoids claims by the customer of unsatisfactory performance at the conclusion of the project.

The phased approach also allows the developer an opportunity to deal with the customer's changing needs and wants. Few software projects ever completely follow the original specifications. The project usually grows as the work is done and the developer and customer get ideas for a better (and usually more complex) project. Moreover, continuous operating system improvements often guarantee major changes in the course of development,

especially if the project takes more than a year. At the very least, a custom software development project should consist of two phases:

- **Specification phase.** The developer prepares a detailed design specification and proposes a completion schedule. In the model development agreement, the developer includes these items as part of a development plan to be provided to the customer.
- **Development phase.** If the customer is satisfied with the detailed specifications, the customer typically pays a specified sum and the developer commences to create the software.

If the customer is not satisfied with the detailed specifications, the developer either revises them or the contract is canceled. It's usually best for the parties to go their separate ways if they can't reach some sort of agreement up front on detailed project specifications. If the contract is canceled, the developer is still entitled to be paid for the time and expense involved in creating the development plan.

Most custom software projects are too lengthy and complex to be divided into only two phases. After the initial specifications phase is completed, the customer will want the developer to deliver specified portions of the system as they are completed. In some cases, these system portions can run independently of each other so they can be tested as soon as they are completed.

Who Writes the Contract?

We recommend that a developer present the customer with a form agreement to review. The initial draft may undergo many changes, but it serves as the basis for the final contract.



CD-ROM

The full text of a Custom Software Development Agreement is on the forms CD-ROM.

Contents of Custom Software Development Agreements

Open the file now. The sample form contains a number of blank spaces to fill in and several alternative provisions you must choose from. The following section explains each section of the agreement and provides guidance on how to complete it.

Identification of the Parties

This Agreement is made between _____ [NAME OF CUSTOMER] (the "Customer") with a principal place of business at _____ [ADDRESS] and _____ [NAME OF SOFTWARE DEVELOPER] (the "Developer") with a principal place of business at _____ [ADDRESS].

Here at the beginning of the contract, both parties should list their full business names. Later on in the contract, you can use obvious abbreviations. For detailed information on identifying the parties, see Chapter 17.

Purpose of Agreement

Customer desires to retain Developer as an independent contractor to develop the computer software (the “Software”) described in the Functional Specifications contained in Exhibit ____ attached to and made part of this Agreement. Developer is ready, willing, and able to undertake the development of the Software and agrees to do so under the terms and conditions set forth in this Agreement. Accordingly, the parties agree as follows: _____

It is customary to briefly state at the beginning the reason for the contract. This helps establish the parties’ intent to create a legally binding contract, and may be helpful if the contract has to be interpreted and enforced by a judge or arbitrator, perhaps years after it was written. Note that the functional specifications, which describe the software in detail, should be attached to the agreement as an exhibit.

Preparation of Development Plan

Developer shall prepare a development plan ("Development Plan") for the Software, satisfying the requirements set forth in the Functional Specifications. The Development Plan shall include:

- (a) detailed Specifications for the Software;
- (b) a listing of all items to be delivered to Customer under this Agreement ("Deliverables");
- (c) a delivery schedule containing a delivery date for each Deliverable; and

[OPTIONAL: INCLUDE IF DEVELOPER IS PAID FIXED PRICE FOR PROJECT] (d) a payment schedule setting forth the amount and time of Developer's compensation.

Developer shall deliver the Development Plan to Customer by [DEVELOPMENT PLAN DEADLINE]. Customer shall have [NUMBER OF DAYS TO REVIEW] days to review the Development Plan. Upon approval of the Development Plan by Customer, it will be marked as Exhibit ____ and will be deemed by both parties to have become a part of this Agreement and will be incorporated by reference. Developer shall then commence development of Software that will substantially conform to the requirements set forth in the Development Plan.

If the Development Plan is in Customer's reasonable judgment unsatisfactory in any material respect, Customer shall prepare a detailed written description of the objections. Customer shall deliver such objections to Developer within [NUMBER OF DAYS TO OBJECT] days of receipt of the Development Plan. Developer shall then have [NUMBER OF DAYS TO MODIFY] days to modify the Development Plan to respond to Customer's objections. Customer shall have [NUMBER OF DAYS TO RE-REVIEW] days to review the modified Development Plan. If Customer deems the modified Development Plan to be unacceptable, Customer has the option of terminating this Agreement upon written notice to Developer or permitting Developer to modify the Development Plan again under the procedure outlined in this paragraph. If this Agreement is terminated, the obligations of both parties under it shall end except for Customer's obligation to pay Developer all sums due for preparing the Development Plan and the ongoing obligations of confidentiality set forth in the provision of this Agreement entitled "Confidentiality."

[OPTIONAL; FOR USE IF DEVELOPER IS NOT PAID ON HOURLY BASIS] Payment for Development Plan: If the Development Plan is not accepted by Customer and Customer terminates this Agreement, Developer shall be entitled to compensation on a time and materials basis at an hourly rate of \$[HOURLY RATE] plus expenses to the date of termination. Developer shall submit an invoice detailing its time and expenses preparing the Development Plan. If the invoice amount is less than the amounts paid to Developer prior to termination, Developer shall promptly return the excess to Customer. If the invoice amount exceeds the amounts paid to Developer prior to termination, Customer shall promptly pay Developer the difference. [OPTIONAL: "However, Developer's total compensation for preparing the Development Plan shall not exceed \$[AMOUNT]."]

The first phase of the project is the developer's preparation of a software development plan. The development plan should show the customer what the developer intends to do and how long it will take. In this clause, the customer's commitment to proceed with the project is contingent upon its acceptance of the development plan. This protects both the customer and developer. The customer's financial exposure is minimized: If the design fails to meet its original expectation, the customer can terminate the contract and seek a new developer without much loss. The developer is protected by having the customer agree in writing to the detailed specifications and delivery schedule before it actually starts to create the software. This prevents later claims that the finished product doesn't meet the customer's requirements.

This provision also gives the developer time to cure any deficiencies in the plan before the customer may terminate the contract. This seems only fair given the substantial investment of time and effort the developer will usually expend to create a development plan.

Following are the key points the agreement addresses:

Specifications

Before the development agreement is signed, the customer (often with the aid of a computer consultant), or the developer, should create a functional specification describing the software in general terms.

The developer will use the functional specifications as a guide in creating a far more technical detailed specification. The detailed specifications serve as the final project blueprint. They describe the project in a very precise technical manner and should provide the developer with all the information it needs to actually create the software.

It can be difficult or impossible for a technically unsophisticated customer to get a feel for how the proposed software will function from written specifications. For this reason, the developer may be asked to create a prototype of the actual system. The prototype should include a complete set of proposed report formats, screen displays, and menus so that the customer can review the program's customer interface. The prototype can be done either by hand on "storyboards," or on computer, using readily available software.

Deliverables

The development plan should list all deliverables—that is, every item the customer wants the developer to provide. This includes not only the software itself, but the program and customer documentation as well.

Following is a checklist of deliverables for a typical software project:

- final specifications for the completed software, showing any changes from the specifications contained in the Development Plan;

- complete customer documentation, including a description of how to access and use each application, screen prints of menus and input/output screens, data input descriptions, sample output/report forms, error code descriptions and solutions where appropriate, and explanations of all necessary disks and data used by the software;
- complete program/technical documentation, including technical information about files and their locations, file names, file/database structure, record structure, and layout and data elements;
- if the customer will receive a copy of the source code, program source code listings with comments;
- description of backup and recovery procedures, including process, medium for backup, and number of diskettes or tapes to do a complete backup; and
- master copy of software on magnetic media, including all programs, online documentation, and any documentation developed on computer.

Delivery schedule

The development plan should also include a schedule showing when each deliverable will be completed. The developer must take care that the delivery schedule provides sufficient time to complete the project.

Developers tend to be incorrigible optimists, so they constantly underestimate how long development projects will take. It's best for the developer to be realistic up front, and not promise the customer what it can't deliver. Since delays are so common, the development agreement should state how delays will be dealt with.

Payment for development plan

Drafting a development plan—particularly the specifications—is difficult and time-consuming. Thus, the developer should be compensated for its efforts fairly. Where the developer is being paid a fixed price, it normally receives a portion as a down payment upon execution of the contract. If the plan is not accepted and the contract is canceled, the down payment may provide the developer with too little or too much money for the work it did on the plan.

This clause deals with this problem by providing that the developer will be paid on a “time and materials” basis for the development plan if the plan is not accepted and the contract is canceled. (“Time and materials” means the developer is paid for its time and actual costs.) If this amount is less than the down payment, the developer must return the excess to the customer. This paragraph need not be included in contracts in which the developer is to be paid on an ongoing hourly basis. It would merely be redundant.

Payment

There are two basic ways to pay a developer for creating custom software:

- a pay per hour (“time and materials”) agreement; or
- a fixed price agreement.

Alternative 1: “time and materials” agreements

Developer shall be compensated at the rate of \$[RATE] per hour [OR “day,” “week,” “month”]. Payment will be made within [NUMBER OF DAYS] days of Developer’s submission of an invoice for work completed. [OPTIONAL: “Unless otherwise agreed upon in writing by Customer, Customer’s maximum liability for all services performed during the term of this Agreement shall not exceed \$[MAXIMUM AMOUNT].”]

Under a “time and materials” agreement, the developer charges the customer by the hour, day, or month at a flat hourly cost. This is by far the safest way to be paid if it’s difficult to estimate exactly how long a project will take. However, many clients will wish to place a cap on your total compensation because they’re afraid you’ll “pad” your bill. If this is the case, include the optional sentence providing for a cap, but make sure it’s large enough to allow you to complete the job.

Alternative 2: fixed price agreements

The total contract price shall be set forth in the Development Plan. Customer shall pay the Developer the sum of \$[INITIAL AMOUNT] upon execution of this Agreement and the sum of \$[AMOUNT IF PLAN APPROVED] upon Customer’s approval of the Development Plan. The remainder of the contract price shall be payable in installments according to the payment schedule to be included in the Development Plan.

Each installment shall be payable upon completion of each project phase by Developer and acceptance by Customer in accordance with the provision of this Agreement entitled “Acceptance Testing of Software.”

The other payment option is for the customer to pay the developer a fixed price for the entire project. In theory, this payment scheme favors the customer by giving the customer certainty as to what the project will cost. Moreover, if payments are tied to the progress of the developer’s work, it gives the customer substantial leverage to insist on timely and successful completion of the project.

At first glance, fixed-price agreements would seem to be risky for developers: If the project takes much longer than originally anticipated, the developer could end up losing money. However, as a practical matter, fixed-price agreements usually do not end

up favoring the customer as much as one would think. If it turns out that the fixed-price originally agreed upon will not provide the developer with fair compensation, because the project ends up taking too long, the customer will probably end up agreeing to pay the developer more money. Otherwise the developer may quit or end up delivering a hastily completed and shoddy product.

The developer's fixed-price quote should be included in the Development Plan, which will also contain the detailed specifications, deliverables list, and delivery schedule. Obviously, the developer should complete these last three items, particularly the detailed specification, before deciding how much to charge for the entire project. Only at this point can the developer have a reasonably accurate idea of how much time the project will require.

Fixed-price contracts are normally paid in installments, with payment of each installment tied to completion and acceptance of a phase of the project. In addition, an initial down payment is usually made when the contract is signed. This provision requires that a payment schedule be included as part of the development plan, which will be attached to the contract as an exhibit. The developer should make sure that the schedule requires the customer to make regular periodic payments so that the developer can meet its own financial obligations.

The amount of the down payment is often a point of contention in fixed-price contracts. Naturally, the developer should obtain as large a down payment as possible.

Payment of Developer's Costs

Whether a fixed-price or time and materials payment arrangement is used, the developer is usually reimbursed by the customer for at least some out-of-pocket expenses incurred in performing its duties under the contract. The extent of such reimbursement is a matter for negotiation.

Alternative 1: full reimbursement:

Customer shall reimburse Developer for all out-of-pocket expenses incurred by Developer in performing services under this Agreement. Such expenses include, but are not limited, to:

- (a) all communications charges;
- (b) costs for providing conversion services for converting Customer's database;
- (c) media costs;
- (d) travel expenses other than normal commuting, including airfares, rental vehicles, and highway mileage in company or personal vehicles at ____ cents per mile; and
- (e) other expenses resulting from the work performed under this Agreement.

Developer shall submit an itemized statement of Developer's expenses. Customer shall pay Developer within 30 days from the date of each statement.

The first alternative calls for payment of all the developer's out-of-pocket expenses;

this may include items like communications charges (telephone, fax, postage, etc.).

Alternative 2—Partial reimbursement: One problem with the first alternative is that many government agencies consider payment of a worker's ordinary business expenses to be a mild indicator of an employment relationship.

Alternative 2: partial reimbursement

Customer shall reimburse Developer for all reasonable travel and living expenses necessarily incurred by Developer while away from Developer's regular place of business and engaged in the performance of services under this Agreement.

The second alternative clause limits reimbursement to extraordinary expenses, such as nonlocal travel, and will not be viewed as indicative of an employee-employer relationship.

Late Fees

Late payments by Customer shall be subject to late penalty fees of _____% per month from the due date until the amount is paid.

Many developers charge a late fee if the customer doesn't pay within the time

specified in the development agreement or invoice. Charging late fees for overdue payments can get customers to pay on time. The late fee is normally expressed as a monthly interest charge—for example, 1% per month.

If you wish to charge a late fee, make sure it's mentioned in your agreement. You should also clearly state what your late fee is on all your invoices.

Materials

Customer shall make available to Developer, at Customer's expense, the following materials, facilities, and equipment:

_____ [LIST].

These items will be provided to Customer by _____ [DATE].

If the client will provide you with any materials or equipment, list them in this optional clause. If you need these items by a specific date, specify the deadline as well.

Changes in Project Scope

If at any time following acceptance of the Development Plan by Customer, Customer should desire a change in Developer's performance under this Agreement that will alter or amend the Specifications or other elements of the Development Plan, Customer shall submit to Developer a written proposal specifying the desired changes.

Developer will evaluate each such proposal at its standard rates and charges. Developer shall submit to Customer a written response to each such proposal within ten working days following receipt thereof. Developer's written response shall include a statement of the availability of Developer's personnel and resources, as well as any impact the proposed changes will have on the contract price, delivery dates, or warranty provisions of this Agreement.

Changes to the Development Plan shall be evidenced by a "Development Plan Modification Agreement." The Development Plan Modification Agreement shall amend the Development Plan appropriately to incorporate the desired changes and acknowledge any effect of such changes on the provisions of this Agreement. The Development Plan Modification Agreement shall be signed by authorized representatives of Customer and Developer, whereupon Developer shall commence performance in accordance with it.

Should Developer not approve the Development Plan Modification Agreement as written, Developer will so notify Customer within ten working days of Developer's receipt of the Development Plan Modification Agreement. Developer shall not be obligated to perform any services beyond those called for in the Development Plan prior to its approval of the Development Plan Modification Agreement.

For purposes of this Agreement, each Development Plan Modification Agreement duly authorized in writing by Customer and Developer shall be deemed incorporated into and made part of this Agreement. Each such Development Plan Modification Agreement shall constitute a formal change to this Agreement adjusting fees and completion dates as finally agreed upon.

It's quite common in the course of software development for the customer to wish to make changes—adding new features and/or deleting others. Of course, any changes to the specifications or any other provision of the contract should be in writing and signed by both parties to be effective.

Major changes in the scope of the project may greatly increase—or decrease—the amount of work the developer has to do. If the developer is paid a fixed fee, it is only fair that its compensation should be increased or decreased to reflect the change in the workload. This usually is not an issue where the developer is paid by the hour;

but if there is a cap on the total hourly compensation to be paid, the cap may have to be increased.

Changes in the scope of the project may also require changes in proposed delivery dates, and perhaps in the contract's warranty provisions.

This provision sets forth a procedure for the parties to follow if the customer wants to change the specifications or other parts of the development plan. First, the customer must submit a written proposal to the developer showing the desired changes. The developer then responds to this proposal in writing, stating, among other things, what impact the desired changes will have on the contract price and delivery schedule. If the customer wants to go ahead with the changes, it submits a "Development Plan Modification Agreement" specifying all the agreed-upon changes and their effect on the provisions of the Agreement. The developer has ten days to accept or reject the Development Plan Modification Agreement.

In many instances, the developer, not the customer, will wish to alter the specifications. This may be because the specifications contain an error, do not accurately reflect the customer's true needs, or the project as originally agreed upon proves commercially unfeasible for a developer being paid a fixed fee. All such changes should be in writing and signed by both parties as required by the clause dealing with modifications to the agreement.

Delays

Developer shall use all reasonable efforts to deliver the Software on schedule. However, at its option, Developer can extend the due date for any Deliverable by giving written notice to Customer. The total of all such extensions shall not exceed ____ [NUMBER] of days.

Any delay or nonperformance of any provision of this Agreement caused by conditions beyond the reasonable control of the performing party shall not constitute a breach of this Agreement, provided that the delayed party has taken reasonable measures to notify the other of the delay in writing. The delayed party's time for performance shall be deemed to be extended for a period equal to the duration of the conditions beyond its control.

Conditions beyond a party's reasonable control include, but are not limited to, natural disasters, acts of government after the date of the Agreement, power failure, fire, flood, acts of God, labor disputes, riots, acts of war, and epidemics. Failure of subcontractors and inability to obtain materials shall not be considered a condition beyond a party's reasonable control.

It is rare for custom software to be completed and delivered by the developer exactly on schedule. There are many reasons for this. Estimating how long it will take to create new software is an uncertain art at best (exacerbated by the fact that most

programmers are incorrigible optimists and many customers are terminally impatient). Delays may also result if the developer has staffing changes or increased demands on existing staff.

The first paragraph of this section is designed to give the developer some leeway in the delivery schedule. Any deadline can be extended by the developer by giving the customer written notice. However, a cap is placed on the total of all the extensions. You need to decide how many days this cap should be.

Sometimes a project is delayed due to circumstances beyond the developer's control—for example, where an earthquake, fire, or other act of God destroys or severely damages the developer's office and equipment. In this event, it is only fair that the developer's delay be excused. A special provision (often called a “force majeure” clause) is commonly included in development contracts to excuse a party's nonperformance due to circumstances beyond its control. The second and third paragraphs of this section contain such a provision.

Acceptance Testing of Software

“Acceptance testing” is a procedure by which the software is tested to see if it satisfies the detailed specifications set forth in the Development Plan. Acceptance testing is one of the most important phases of the software development process. The purpose

is to determine whether the software does what it is supposed to do and is reliable. Particularly where safety is involved (for example, software implemented in a “911” service or designed to run an elevator), the software should be tested as thoroughly as possible.

The acceptance testing should force the software to perform repeatedly, without failure, on a variety of the customer's actual data, with speed and accuracy to match the specifications. It is also a good idea to stress-test a system—that is, try to break it—to see how it recovers. Each section of the software should be tested independently and in combination with other sections.

The nature of the tests to be performed, the data to be used, and the procedure to be followed should be defined by the parties before the testing begins. In many cases, the acceptance tests cannot be well defined until the final specifications are agreed upon or even until the software is completed. The specifications for acceptance testing should be included as part of the development plan. Of course, these specifications may be subsequently modified by the parties.

Unfortunately, many bugs will not be discovered during acceptance testing. Some bugs appear only after many hundreds or thousands of customer hours. For this reason, it's common for the developer to agree to fix bugs free of charge for a stated period after the software is accepted—usually 90 days to a year.

Trivial Bugs

No software is absolutely perfect. It is impossible for a developer to create software of any complexity that is completely bug-free. However, there is a big difference between true bugs—defects that prevent the software from performing the customer's required tasks satisfactorily—and interface and peripheral glitches and other minor irritants that don't materially affect the software's performance. A developer should never promise that its software will be completely bug-free. Moreover, the acceptance criteria should be designed so as to overlook trivial bugs that can simply be ignored.

Alternative 1: acceptance testing provision for multiphase projects

Immediately upon completion of each development phase set forth in the Development Plan's delivery schedule, Developer shall deliver and install the Software and shall deliver all documentation and other materials required to be provided in accordance with the delivery schedule. Customer shall have [NUMBER OF TESTING DAYS] days from the delivery of the Software to inspect, test, and evaluate it to determine whether the Software satisfies the acceptance criteria in accordance with procedures set forth in the Development

Plan, or as established by Developer and approved by Customer prior to testing.

If the Software does not satisfy the acceptance criteria, Customer shall give Developer written notice stating why the Software is unacceptable. Developer shall have 30 days from the receipt of such notice to correct the deficiencies. Customer shall then have 30 days to inspect, test, and reevaluate the Software. If the Software still does not satisfy the acceptance criteria, Customer shall have the option of either: (1) repeating the procedure set forth above, or (2) terminating this Agreement pursuant to the section of this Agreement entitled "Termination." If Customer does not give written notice to Developer within the initial 30-day inspection, testing, and evaluation period or any extension of that period, that the Software does not satisfy the acceptance criteria, Customer shall be deemed to have accepted the Software upon expiration of such period.

Upon completion of the final development phase set out in the Development Plan, acceptance testing shall be performed on the Software in its entirety to determine whether the Software satisfies the acceptance criteria and operates with internal consistency. Customer shall have [NUMBER OF TESTING DAYS FOR FINAL PRODUCT] days to perform such tests. If the completed Software does not satisfy the acceptance criteria, the parties shall follow the acceptance procedures described in the preceding paragraph [OPTIONAL: except that the time periods for corrections,

inspection, reevaluation, and notice shall be increased to ____ [NUMBER] days] [END OF OPTION].

[INCLUDE IN FIXED PRICE CONTRACTS: If and when the acceptance tests establish that the Software delivered upon completion of any phase of development complies with the acceptance criteria, Customer shall promptly notify Developer that it accepts the delivered Software.] [END ALTERNATIVE 1]

Complex software systems often consist of a number of independent units or “modules” that can be tested when completed. This way, the customer doesn’t have to wait until the entire project is finished to test the software and see how the work is progressing. Of course, the schedule for delivery of each portion of the system should be set forth in detail in the development plan’s delivery schedule.

If the software does not satisfy the acceptance criteria, the customer should explain the problems to the developer and then give it an opportunity to correct them. Developers are typically given 30 days to make corrections, but more time may be required if the software is extremely complex. If the developer cannot make the corrections, or redelivers software that is still nonconforming, the customer should have the option of giving the developer more time or terminating the contract.

When all phases of development are completed, the software should be tested in its entirety. The same acceptance procedure is used as for testing each phase, except that optional language allows the time periods for the developer’s corrections and the customer’s review and reevaluation to be increased beyond the 30 days allowed for acceptance of each phase; 30 days may simply not be enough time to test and evaluate the entire program.

Alternative 2: testing when project completed

Customer shall have 30 days from the date of delivery of the Software in final form to inspect, test, and evaluate it to determine whether the Software satisfies the acceptance criteria in accordance with procedures set forth in the Development Plan, or as established by Developer and approved by Customer prior to testing.

If the Software does not satisfy the acceptance criteria, Customer shall give Developer written notice stating why the Software is unacceptable. Developer shall have 30 days from the receipt of such notice to correct the deficiencies. Customer shall then have 30 days to inspect, test, and evaluate the Software. If the Software still does not satisfy the acceptance criteria, Customer shall have the option of either (1) repeating the procedure set forth above, or (2) terminating this Agreement pursuant to the section of this Agreement entitled "Termination." If Customer does not give written notice to Developer within the initial 30-day inspection, testing, and evaluation period or any extension of that period, that the Software does not satisfy the acceptance criteria, Customer shall be deemed to have accepted the Software upon expiration of such period.

Some types of software really can't be tested until the entire system is completed and running, except perhaps the print and

display functions. This clause is a simpler acceptance testing provision designed to be used where serious testing won't be conducted until the software is delivered in final form. This provision should also be used for projects where the entire program will be delivered at one time, rather than in phases.

Training

Developer shall provide [NUMBER OF TRAINING DAYS] days of training in the use of the Software by at least one (but not more than [MAXIMUM NUMBER OF TRAINERS]) qualified Developer personnel ("trainers"). The training will be conducted on such dates and locations as the parties may agree.

Customer will be responsible for all costs and expenses of all Customer's trainees, including room, board, transportation, salary, insurance and other benefits, and other expenses while attending the training.

[OPTIONAL:] Customer shall pay Developer the sum of \$[AMOUNT] for each [HOUR/DAY] of training by each trainer, plus each trainer's travel expenses. [END OPTION 2]

The customer and/or its employees will usually have to be trained how to use the software. This provision requires the developer to conduct training. Where the developer is paid a fixed fee, the cost of training can be included in the overall fee.

However, where the developer is paid on an hourly basis, it will have to be paid extra for training. In either event, the customer should be responsible for its own costs associated with such training.

Maintenance of Software

Beginning on the first day of the first month following expiration of the warranty period set forth in the section of this Agreement entitled "Warranties," Developer shall provide the following error-correction and support services:

- (a) telephone hot-line support during Developer's normal days and hours of business operation. Such support shall include consultation on the operation and utilization of the Software. Customer shall be responsible for all telephone equipment and communication charges related to such support; and
- (b) error-correction services, consisting of Developer using all reasonable efforts to design, code, and implement programming changes to the Software, and modifications to the documentation, to correct reproducible errors therein so that the Software is brought into substantial conformance with the Specifications.

Payment for Maintenance: Customer shall pay Developer for error-correction and support services the annual sum of \$(MAINTENANCE AMOUNT), payable in quarterly installments beginning on the first day of the first month following expiration of any warranty period. Three years after the date of Customer's final acceptance of the Software, Developer shall be entitled to increases in the maintenance fee upon at least ten days' prior written notice to Customer.

Customer's Role in Maintenance: The provision of the error-correction and support services described above shall be expressly contingent upon Customer promptly reporting any errors in the Software or related documentation to Developer in writing and not modifying the Software without Developer's written consent.

Term of Support: Subject to timely payment by Customer of the maintenance fees, Developer shall offer the maintenance described above for a minimum of [NUMBER OF YEARS] years after completion of the development work under this Agreement.

Customer Termination of Maintenance: Customer may discontinue the maintenance services described above upon not less than 90 days' written notice to Developer.

“Maintenance” is a term used for upkeep of software after it has been put into operation. There are two types of maintenance:

- fixing bugs (sometimes called “remedial maintenance”); and
- modifying and/or enhancing software because the customer’s needs have changed (sometimes called “adaptive maintenance”).

It can cost far more to maintain software than to develop it in the first place. Many developers consider maintenance to be a nuisance and would prefer to use their resources on other new development projects. Such developers want to terminate their work for the customer upon delivery and acceptance of the software. The customer, on the other hand, needs someone to maintain the software. It is often most efficient for the persons who originally create a piece of software to maintain it. Their familiarity with the original software’s design should enable them to do a better job than someone who is completely new to the software.

Typically, custom software development agreements contain a warranty in which the developer promises to fix any bugs for free for a limited time—usually 90 days to a year. The following provision addresses post-warranty remedial maintenance (bug correction) only. Adaptive maintenance (enhancements and modifications) should be handled in a separate agreement.

This provision requires the developer to perform remedial maintenance and provide telephone hotline support after the warranty expires. The customer pays the developer

an annual fee for this service. Note that the provision applies only to “reproducible errors ... so that the Software is brought into substantial conformance with the Specifications.” This protects the developer from committing to fix trivial errors or errors that may occur only once in many thousands of hours of operation and that cannot be readily reproduced.

Ownership of Software

The moment computer code is written, it is protected by copyright and someone becomes the owner of that copyright. Similarly, patent and trade secret ownership rights may come into existence. Many software developers and customers harbor the misapprehension that, since the developer is being paid to create the software by the customer, the customer will automatically own all rights to it. This is not the case. Absent a written contract transferring ownership from the developer to the customer, the developer will most likely own the copyright in the software (unless the developer is considered the customer’s employee or, perhaps, if it was part of a larger work and was prepared under a written work for hire agreement).

One of the most important functions of a software development agreement is to establish exactly who will own the intellectual property rights to the software to be created. This is often one of the most hotly contested issues between the developer and customer, and can easily become a “deal breaker.” Ownership rules

for intellectual property are covered in detail in Chapter 11. This section focuses on how to handle these issues in a custom software development agreement.



CAUTION

Ownership Isn't Intellectual

Property Ownership. Owning the material object in which software is embodied—whether a CD-ROM, hard disk, or other medium—is separate and distinct from intellectual property ownership. When a customer is given a disk containing a software program written by the developer, the customer owns that disk—no one can legally take it away from him—but the customer cannot exercise any of the developer's exclusive copyright rights in the software contained on that disk absent a transfer (license or sale) of such rights from the developer to the customer.

This means that the customer could not legally make more than one archival copy of the software, could not distribute it to the public, and could not create derivative works based upon it—modified versions, for example. This is what we're talking about when we discuss software ownership—ownership of intangible intellectual property rights—not ownership of disks or other media.

Ownership issues are critically important for the developer. If a developer signs a contract with a customer transferring its ownership of the intellectual property rights in the software, the developer can never use, sell, or license that software again. A developer who signs such a contract may end up giving up far more than it actually

realized or intended—it may lose the ability to do similar work for other customers in the future.

We emphasized the “may” above because no one knows exactly to what extent computer software is protected by the law. It is clear that copyright protects source code, so a developer who relinquishes copyright ownership in a program may not use that same source code again unless provisions for such use are made as part of the transaction (or in a later license). But copyright protection might go beyond a program's literal source code—for instance, to the structure, sequence and organization of the code.

The potential problems do not end here. Copyright does not protect ideas, concepts, know-how, techniques, formulae, or algorithms. However, these items may be protected by trade secrecy or, in some instances, patents. Thus, a developer that relinquishes all its intellectual property rights in the software it creates for a customer could conceivably be barred by trade secrecy or patent law from using similar ideas, techniques, and so forth in other programs.

The moral is this: A developer should make sure that it either retains enough ownership rights in what it creates to allow it to continue to do similar development work for other customers, or obtains enough money from the customer to compensate it for the possible lost future business.

One way a developer can protect its future business is to obtain a nonexclusive

license to use the software. Another is to retain ownership of the software and provide the customer with a license to use the software. Both these options are discussed below. In addition, the developer should be careful to retain ownership of its “background technology.” That includes the programs and materials that the developer uses over and over again in most or all of its projects—such as routines for displaying menus, document assembly, and printing.

Four ownership alternatives are provided:

- sole ownership by the customer;
- ownership by developer with an exclusive license to customer;
- ownership by developer with a nonexclusive license to customer; and
- joint ownership.

Alternative 1: ownership by customer

Developer assigns to Customer its entire right, title, and interest in anything created or developed by Developer for Customer under this Agreement (“Work Product”), including all patents, copyrights, trade secrets, and other proprietary rights. This assignment is conditioned upon full payment of the compensation due Developer under this Agreement.

Developer shall execute and aid in the preparation of any documents necessary to secure any copyright, patent, or other intellectual property rights in the Work Product at no charge to client. However, Customer shall reimburse Developer for reasonable out-of-pocket expenses.

[OPTIONAL:] Customer grants to Developer a nonexclusive, [CHOOSE ONE: “irrevocable license” OR “license for the term of [NUMBER OF YEARS] years”) to use the Work Product. [ADD ANY PAYMENT PROVISIONS OR OTHER RESTRICTIONS]
[END ALTERNATIVE 1]

Because they are paying the developer to create the software, some customers insist on receiving sole ownership. The following clause grants all ownership rights in the software to the customer, with no restrictions. Typically, a developer that relinquishes all its ownership rights will demand more payment than if it were allowed to retain at least some ownership

and profit from the software by licensing it to others.

As mentioned above, an assignment of all of the developer's rights means that the developer may not use the software created for the customer, for example, by including part of it in a program created for someone else. This can create a substantial hardship for the developer, especially one that regularly works on similar types of software.

One way a developer can make such an ownership transfer more palatable is to have the customer grant the developer back a nonexclusive license to use the software. A nonexclusive license gives someone the right to use software or other copyrighted work, but does not prevent the copyright owners from granting others the same rights at the same time.

The license can be limited in any way—such as in duration, or in the type of software in which it can be used. For example, the developer can be barred from using the software to help the customer's named competitors or from developing similar types of products. The optional provision granting a license to the developer would be used in addition to the assignment from developer to customer.

Alternative 2: ownership by developer with exclusive license to customer

Developer shall retain all copyright, patent, trade secret, and other intellectual property rights Developer may have in anything created or developed by Developer for Customer under this Agreement ("Work Product"). Developer grants Customer a nontransferable license to use the Work Product. The license is conditioned upon full payment of the compensation due Developer under this Agreement.

The license shall be exclusive in [NAME TERRITORIES, SUCH AS "the United States"] for a period of [TIME PERIOD] following acceptance by Customer of the Software as set forth in this Agreement. The license shall automatically revert to a perpetual nonexclusive license following the period of exclusivity.

The license shall authorize Customer to:

- (a) install the Software on computer systems owned, leased, or otherwise controlled by Customer;
- (b) utilize the Software for its internal data-processing purposes (but not for time-sharing or service bureau purposes); and
- (c) copy the Software only as necessary to exercise the rights granted in this Agreement.

Another option is for the developer to retain ownership of the software and give the customer an exclusive license to use it. (Nonexclusive licenses are covered below, in Alternative 3.)

With an exclusive license, only the customer has the right to use it—within the scope of the license. If an exclusive license gives the customer the right to use the software in every possible context at every possible location, it would be the functional equivalent of ownership. In practice, however, the parties usually agree to limit the customer's use rights. For example, the customer's right to use the software may be limited as to duration, area (worldwide or domestic), market, or hardware (the customer could be permitted to use the software only on a particular platform). Under this type of license, the developer would have the exclusive right to modify the software and could sell or license it to others outside the customer's area of exclusivity.

This type of arrangement often benefits both the customer and developer: The customer is assured that the developer will not sell or license the software to competitors during the term of the exclusive license. At the same time, the developer retains control over the software and will have the opportunity to earn income by licensing it to others outside the area of the customer's exclusivity and/or after the exclusive license expires.

This provision grants an exclusive license that is limited as to time. When the exclusive license expires, the customer receives a perpetual nonexclusive license,

meaning that the developer is free to license the software to others. This license is nontransferable—it does not permit the customer to sublicense the software to others, but this can be permitted if the parties desire. The license permits the customer to use the software on any number of computers for internal purposes and to make as many backup copies as it needs.

Alternative 3: ownership by developer with nonexclusive license to customer

Developer shall retain all copyright, patent, trade secret, and other intellectual property rights Developer may have in anything created or developed by Developer for Customer under this Agreement ("Work Product"). Subject to payment of all compensation due under this Agreement, Developer grants Customer a nonexclusive, nontransferable, royalty-free license to use the Work Product.

The license shall authorize Customer to:

- (a) install the Software on computer systems owned, leased, or otherwise controlled by Customer;
- (b) utilize the Software for its internal data-processing purposes (but not for time-sharing or service bureau purposes); and
- (c) copy the Software only as necessary to exercise the rights granted in this Agreement.

The most favorable ownership arrangement for the developer may be for the customer to be given only a nonexclusive license to use the software. This means that the developer is free to license the software to anyone else, including the developer's competitors. This type of ownership arrangement would likely result in the lowest possible price to the customer, because the developer may earn additional income by licensing the software to others.

Alternative 4: joint ownership

Developer hereby grants Customer an undivided one-half interest in the Software and associated documentation. The Software may be freely used by either party without accounting to the other party. Customer and Developer agree to execute all documents reasonably necessary to legally establish their joint ownership of the Software.

Yet another option is for the customer and developer to jointly own the software. Under a joint ownership arrangement, each party is free to use the software or grant nonexclusive licenses to third parties without the other's permission (unless they agree to restrict this right). Normally, joint owners must account for and share with each other any monies they earn from granting such licenses. This is probably not desirable in the developer-customer situation, so this provision specifically

provides that neither party need account to the other—in other words, they need not share any money they earn from the software.

Ownership of Background Technology

Customer acknowledges that Developer owns or holds a license to use and sublicense various preexisting development tools, routines, subroutines, and other programs, data, and materials that Developer may include in the Software developed under this Agreement. This material shall be referred to as "Background Technology." Developer's Background Technology includes, but is not limited to, those items identified in Exhibit ____, attached to and made a part of this Agreement.

Developer retains all right, title, and interest, including all copyright, patent rights, and trade secret rights in the Background Technology. Subject to full payment of the consulting fees due under this Agreement, Developer grants Customer a nonexclusive, perpetual worldwide license to use the Background Technology in the Software developed for and delivered to Customer under this Agreement, and all updates and revisions thereto. However, Customer shall make no other commercial use of the Background Technology without Developer's written consent.

A software developer will normally bring to the project various development tools, routines, subroutines, and other programs, data, and materials. One term for these items is “background technology.” It’s quite possible that background technology may end up in the final product. For example, this may include code used for installation, window manipulation, displaying menus, data searching, data storing, and printing.

If the developer transfers complete ownership of the software to the customer, the customer also may end up owning this background technology. Such an arrangement would prohibit the developer from using the background technology in other projects without obtaining the customer’s permission (and perhaps paying a fee). A developer is usually well-advised to avoid this problem by making sure the agreement provides that the developer retain all ownership rights in background technology. In this event, the agreement also should give the customer a nonexclusive license to use the background technology that’s included in the customer’s software.

This provision permits the customer to use the background technology as included in the software, but keeps ownership in the hands of the developer. The developer should prepare a separate exhibit that identifies in as much detail as possible the background technology to be included in the software.

This provision need not be included in a contract in which the developer retains ownership of the software.

Identify Background Technology

If possible, identify your background technology in the source code copies of the programs in any printouts of code you deliver to the customer. You might include a notice like the following where such material appears: “[Your Company Name] CONFIDENTIAL AND PROPRIETARY.”

Source Code Access

Customer agrees that the Software developed under this Agreement shall be delivered to Customer in object code form only. Developer agrees that one copy of the source code version of the Software and associated documentation shall be deposited with an escrow agent specializing in software escrows to be mutually agreed upon in writing by Developer and Customer after good faith negotiation. Customer and Developer shall enter into a supplementary escrow agreement with the escrow agent.

The source code shall be delivered to the escrow agent within [NUMBER OF DAYS] days after delivery of the object code to Customer. Thereafter, the source code version of all updates, enhancements, and modifications of the Software created by Developer on Customer's behalf, as well as associated documentation, shall be deposited by Developer with the escrow agent. Customer shall pay all fees necessary to establish and maintain the escrow.

Developer hereby grants to Customer a contingent license to receive the source code from the escrow agent and to use the source code to support its use of the Software in machine-readable form if one or more of the following conditions occurs:

- (a) Developer, whether directly or through a successor or affiliate, ceases to be in the software business;
- (b) Developer fails to fulfill its obligations to maintain the Software as provided in this Agreement;
- (c) Developer becomes insolvent or admits insolvency or a general inability to pay its debts as they become due; or
- (d) Developer files a petition for protection under the U.S. Bankruptcy Code, or an involuntary petition is filed against it and is not dismissed within 60 days.

[OPTIONAL:] (e) Developer comes under the control of a competitor of Customer. [END OPTION]

The source code shall be used solely by Customer to maintain the Software and shall be subject to every restriction on use set forth in this Agreement. Customer agrees not to disclose the source code to third parties except on a need-to-know basis under an appropriate duty of confidentiality.

If the customer obtains ownership of the software, it should receive the source code and system/program documentation created by the developer. But if the developer retains ownership and merely grants the customer a license to use the software,

source code access can become an important issue.

Some clients insist on having access to the developer's source code. They're afraid the original developer won't be around months or years down the line when the

software needs to be maintained, modified, or enhanced. They're afraid it will be difficult or impossible to do such work without the source code. However, as a practical matter, access to the source code is greatly overrated as a benefit for a software customer. The fact is, even with decent documentation, the expense involved in picking up somebody else's code is often greater than programming from scratch.

Software developers are understandably reluctant to give a customer a copy of their proprietary source code. A software developer's most important asset is usually its source code, which may contain highly valuable trade secrets. One way for a developer to deal with a client who demands access to the source code is to agree to a source code escrow.

Under a source code escrow agreement, the developer gives a copy of the source code and documentation to a neutral third party for safekeeping. The third party will release the source code to the customer only upon the occurrence of specified conditions, such as the developer's bankruptcy or failure to maintain the software. This keeps the developer's source code confidential while, in theory, assuring the customer access to it should it become necessary.

So long as the customer pays for the escrow and the conditions for its release are reasonable, this arrangement works well for the developer. It avoids giving the customer the source code outright.

Literally anybody can serve as the escrow agent. In the past, attorneys, accountants, and bank escrow departments have

frequently been used. In recent years, though, a number of houses specializing in software escrows have been established. You can find a list of these on the Yahoo.com Internet directory at http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Computers/Services/Source_Code_Escrow. You may be able to find others by doing an Internet search.

Using an escrow firm that specializes in software affords several advantages. First, these companies provide a carefully controlled environment for storage of magnetic media, assuring that the deposited material is undamaged. In addition, some software escrow houses also provide a verification service. This may consist of simply making sure the materials deposited match the requirements of the escrow agreement. However, for an additional charge, some escrows will take the source code provided by the developer and compile it (turn it into object code) to see if it is identical to the object code given to the customer. Software escrow companies normally supply detailed form escrow agreements for the parties to sign. These agreements attempt to evenly balance the interests of both parties.

Since there is no guarantee when a development project is commenced that the software will be satisfactorily completed, it makes sense to delay spending the time and money involved in setting up a source code escrow until the software is actually written. This optional clause obliges the parties to set up an escrow with a software escrow house to be determined later. The escrow agreement is also to be negotiated later.

This clause makes clear that the customer will pay for the escrow. This is fair because the escrow is really for the customer's benefit. It also sets forth the circumstances under which the escrowed materials will be released to the customer by the escrow company; this should also, of course, be spelled out in detail in the escrow agreement.

Warranties

A warranty is a promise or statement regarding the quality, quantity, performance, or legal title of something being sold. We all have some familiarity with warranties. Whenever we buy an expensive product from a car, to a television, to a computer, the seller normally warrants that the product will do what it is supposed to do for a specific or reasonable time period, otherwise the seller will fix or replace it.

When goods are sold in the course of business, they generally are warranted. If the goods later prove defective and the seller fails to repair or replace them in accordance with the warranty, it is called a "breach of warranty" and the buyer can seek relief in the courts. In many states, hefty damages can be obtained against sellers who fail to live up to their warranties. These may include not only the cost of replacing the defective software, but any economic losses suffered by the customer as a result of the defect, such as lost profits.

Custom software developers are naturally hesitant about giving a warranty for something that is not yet in existence when the warranty is made. However, some

customers usually demand assurance that the product will work.

Because this is an area of active bargaining between the developer and customer, warranty provisions vary widely. Before we discuss sample warranty provisions for custom software development contracts, let's take a quick look at the various types of warranties.

Express warranties

When a developer makes an actual promise about how the software will work, whether orally or in writing, it is making an "express warranty." An express warranty can be created by using formal words such as "guarantee," "affirm," or "warrant." However, it is important to understand that no magic words are necessary to create an express warranty. Representations made by salespeople, sales literature, statements at product demonstrations, proposals, manuals, or contractual specifications can all constitute express warranties. And this can even be true where the developer did not intend to create a warranty. Express warranties can last for any period of time, ranging from a few months to the lifetime of the software.

Customers often seek an express warranty from software developers guaranteeing that the software is free from defects and will meet the functional and design specifications set forth in the development plan. (See "Alternative 2: providing express warranties," below.) Another common express warranty is a guarantee that the software will not infringe any third party's copyright, patent, or trade secret rights.

Implied warranties

In every commercial transaction involving the sale of movable goods, certain representations by the seller are assumed to be made, even if no words are written or spoken. These representations are implied by state laws based on the Uniform Commercial Code (“U.C.C.”), a set of model laws designed by legal scholars that have been adopted by every state but Louisiana. The U.C.C. establishes uniform rules governing the sale of goods and other commercial transactions. In the past, a number of courts have disagreed on whether custom software qualifies as a “good” governed by these state U.C.C. laws, but today the trend appears to be that these laws apply to custom software sales transactions. (There is no question that the U.C.C. applies when software is bundled with hardware, or when standard off-the-shelf software is customized.)

This book can’t provide a whole course on the U.C.C., but you should be aware that there are four implied warranties that automatically exist in contracts for the sale of goods unless they are expressly disclaimed (disavowed). These are:

- **Implied warranty of title.** All sellers warrant that they are transferring good legal title to the goods; that they have the right to make the transfer; and, as far as they know, the goods are not subject to any liens, encumbrances, or security interests. This warranty is particularly important for software development agreements because the software often contains elements that

have been licensed from or to third parties.

- **Implied warranty against infringement.** The seller warrants that the product will be delivered free of any rightful claim by any third party that the product infringes such person’s patent, copyright, trade secret, or other proprietary rights.
- **Implied warranty of fitness for a specific purpose.** If a customer is relying on the seller’s expertise to select suitable goods, and the seller is aware that the customer intends to use the goods for a particular purpose, the product becomes impliedly guaranteed for that purpose. For example, if a developer knows that the customer needs software to operate an assembly line at a particular speed, and agrees to develop software for that purpose, there is an implied warranty that the software will operate the line at the proper speed even if this specification isn’t made a part of the contract. This warranty almost always applies in custom software transactions.
- **Implied warranty of merchantability.** The seller promises that the goods are fit for their commonly-intended use—in other words, they are of at least average quality. This means that software must perform so as to satisfy most customers’ expectations. The software need only perform in a minimally acceptable manner to satisfy this warranty; it need not satisfy the highest function, speed, or other performance criteria.

Alternative 1: disclaimer of all warranties

[DISCLAIMER OF ALL WARRANTIES] THE SOFTWARE FURNISHED UNDER THIS AGREEMENT IS PROVIDED ON AN "AS IS" BASIS, WITHOUT ANY WARRANTIES OR REPRESENTATIONS EXPRESS, IMPLIED, OR STATUTORY; INCLUDING, WITHOUT LIMITATION, WARRANTIES OF QUALITY, PERFORMANCE, NONINFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. NOR ARE THERE ANY WARRANTIES CREATED BY A COURSE OF DEALING, COURSE OF PERFORMANCE, OR TRADE USAGE. DEVELOPER DOES NOT WARRANT THAT THE SOFTWARE WILL MEET CUSTOMER'S NEEDS OR BE FREE FROM ERRORS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED. THE FOREGOING EXCLUSIONS AND DISCLAIMERS ARE AN ESSENTIAL PART OF THIS AGREEMENT AND FORMED THE BASIS FOR DETERMINING THE PRICE CHARGED FOR THE SOFTWARE.

There is no requirement in most states that the seller of goods provide any warranties at all. The implied warranties discussed above (and any express warranties) may be expressly disclaimed by the seller. This means that the goods are sold "as is." While an "as is" statement will protect a seller from many types of claims, it won't protect it from charges of outright fraud (if it lied about the goods in question). To be effective, "as is" statements must be "conspicuous"—for instance, printed in boldface capitals and large type—so they won't be overlooked by the customer (no proverbial "fine print").

Alternative 2: providing express warranties

(a) **Warranty of Software Performance:** Developer warrants that for [WARRANTY PERIOD] following acceptance of the Software by Customer, the Software will be free from material reproducible programming errors and defects in workmanship and materials, and will substantially conform to the Specifications in the Development Plan when maintained and operated in accordance with Developer's instructions. If material reproducible programming errors are discovered during the warranty period, Developer shall promptly remedy them at no additional expense to Customer. This warranty to Customer shall be null and void if Customer is in default under this Agreement or if the nonconformance is due to:

(1) hardware failures due to defects, power problems, environmental problems, or any cause other than the Software itself;

(2) modification of the Software operating systems or computer hardware by any party other than Developer; or

(3) misuse, errors, or negligence of Customer, its employees or agents in operating the Software.

Developer shall not be obligated to cure any defect unless Customer notifies it of the existence and nature of such defect promptly upon discovery.

(b) **Warranty of Title:** Developer owns and has the right to license or convey title to the Software and documentation covered by this Agreement. Developer will not grant any rights or licenses to any intellectual property or technology that would conflict with Developer's obligations under this Agreement.

(c) **Warranty Against Disablement:** Developer expressly warrants that no portion of the Software contains or will contain any protection feature designed to prevent its use. This includes, without limitation, any computer virus, worm, software lock, drop-dead device, Trojan-horse routine, trap door, time bomb, or any other codes or instructions that may be used to access, modify, delete, damage, or disable Customer's Software or computer system. Developer further warrants that it will not impair the operation of the Software in any way other than by order of a court of law.

(d) **Warranty of Compatibility:** Developer warrants that the Software shall be compatible with the Customer's hardware and software as set forth in the Development Plan Specifications.

THE WARRANTIES SET FORTH IN THIS AGREEMENT ARE THE ONLY WARRANTIES GRANTED BY DEVELOPER. DEVELOPER DISCLAIMS ALL OTHER WARRANTIES EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. [END ALTERNATIVE 2]

Although creating custom software is a difficult and often uncertain process, it is reasonable for the customer to expect the developer to stand behind its product. In a custom software development agreement, the developer typically gives the customer certain express warranties and disclaims any and all implied warranties. The parties need to negotiate exactly what type of express warranties the developer will make. Naturally, the customer wants the developer's warranties to be as expansive as possible, while the developer wishes them to be narrowly drawn. A compromise must be reached.

Alternative 2 contains the following express warranties commonly found in custom development agreements. Include whichever warranties are desired.

- **Warranty of software performance.**

Under a warranty of software performance, the developer guarantees that the software will function properly. The developer promises that the software is free from material defects (but not absolutely perfect) and will perform in substantial conformance with the specifications.

This warranty is usually limited in time, to anywhere from 90 days to one year. During the warranty term, the developer is required to correct any defects and modify the software as necessary, free of charge. Normally excluded from coverage are any defects caused by the customer's misuse of the software or from causes beyond the developer's control, such as a power failure.

- **Warranty of title.** The developer warrants that it has the legal right to grant the customer all rights specified in the contract. This normally means that no intellectual property rights to the software have been licensed to others on an exclusive basis. In addition, if the developer has used any code covered by another's copyright, patent or trade secret protections, it has the legal right to do so.
- **Warranty against disablement.** Developers who are concerned that they might not be paid fully or that the customer may breach the terms of the development contract have been known to include "computer viruses" and disabling devices in their software. These devices are intended to disable the software, either automatically with the passage of time or under the developer's control. The purpose, obviously, is to prevent the customer from using the software if the customer fails to uphold its end of the agreement.

Is this legal? One court has indicated it might be, but only if the developer tells the customer in advance that disabling devices will be included in the software and the customer agrees (*Frank & Sons v. Information Solutions*, (N.D. Okla. No. 88C1474E), *Computer Indus. Lit. Rep.*, Jan. 23, 1989, at 8927-35). In the absence of such notice and consent, a developer who disables a customer's software could be liable for all the customer's resulting damages. These could be substantial,

especially if the customer's entire computer system is affected.

In one well-known case, a developer included a disabling device in an inventory control system it created for cosmetics manufacture for Revlon, Inc. When Revlon stopped paying the developer because it was dissatisfied with the system's performance, the developer sent Revlon a letter warning that it would disable the software if payment was not forthcoming. The developer then activated the disabling device by dialing into Revlon's computer system. As a result, two of Revlon's distribution centers were completely shut down for several days. Revlon estimated its losses at \$20 million. Revlon sued the developer, who settled for an undisclosed amount (*Revlon, Inc. v. Logisticon, Inc.*, No. 705933 (Cal. Super. Ct., Santa Clara Cty., complaint filed Oct. 22, 1990)). The lesson is clear: A developer should absolutely never include any disabling device in software without informing the customer and obtaining its consent in advance. Even then, the developer should think twice if a customer does consent—the customer might sue the developer anyway if activation of a disablement device causes it substantial losses.

Not only will most customers refuse to allow inclusion of a disabling device in their software, they will often demand that the developer expressly guarantee that none has or will be included.

- **Warranty of compatibility.** The warranty of compatibility provides that the software will be compatible with the hardware on which it will run and with any noncustom software included in the customer's system. This warranty is particularly important where a customer is acquiring a system from multiple vendors—that is, hardware from one or more vendors, and software from others.
- **Disclaiming warranties not in agreement.** In return for the express warranties included in the contract, the customer typically agrees to allow the developer to disclaim any and all other warranties, whether express or implied. Such a disclaimer should be typed in capitals, preferably boldface, to be enforceable.

Warranties a Developer Should Not Make

A developer should only provide a warranty as to those matters within its control. This is a matter of fairness and common sense. Matters beyond a developer's control include:

- *Error-free software.* A developer cannot realistically warrant that the software will be completely bug-free. All software, particularly custom software, inevitably contains some bugs. But there is a big difference between true bugs—defects that prevent the software from performing the customer's required tasks satisfactorily—and interface and peripheral glitches and other minor irritants that don't materially affect the software's performance. The most a developer should promise is that the software will not contain "material" defects. A developer should never promise to fix trivial bugs free of charge. Fixing minor bugs can be just as difficult and expensive as fixing major ones. Moreover, the customer really doesn't need to have trivial bugs fixed at all. All

the customer needs is stable software that performs satisfactorily. Trivial bugs that don't prevent the software from performing satisfactorily can simply be ignored.

- *Software will be free from defects in materials and workmanship.* This type of warranty is commonly found in contracts for machinery, including computer hardware. However, it does not belong in software development contracts. Again, no developer can safely guarantee that custom software will be perfect.
- *Software will perform in exact conformance with specifications.* Developers shouldn't promise this unless they're absolutely sure it will be true. It's far better for the developer to warrant that the software will perform "in substantial conformance" with the specifications, meaning that trivial variations will not violate the warranty.

Intellectual Property Infringement Claims

“Intellectual property” is a catch-all term that includes copyrights, patents, trade secrets, and trademarks. (See Chapter 2 for a detailed discussion.) Most software development agreements contain some kind of express warranty against intellectual property infringement. The extent of such a warranty is subject to negotiation.

In this highly litigious world, intellectual property infringement is an issue the developer must think about carefully. In some cases, the developer may even choose to lower its price or make other concessions to the customer to avoid making a broad warranty of noninfringement. Three alternative warranty provisions are provided in the agreement.

Insurance Coverage for Infringement Claims

Your business may be insured for intellectual property infringement claims and you may not even know it. The Comprehensive General Liability Insurance (“CGL”) policies typically obtained by businesses may provide such coverage. Several courts have held that the “advertising injury” provision included in many CGL policies covers infringement claims. However, not all CGL policies provide such

coverage. Ask your insurance broker whether your policy covers infringement claims. If the broker doesn’t know, you may need to consult with an insurance attorney who represents policyholders. If your CGL policy doesn’t cover infringement claims, you may be able to obtain such coverage by purchasing umbrella or excess policy coverage from your insurer.

Alternative 1: limited warranty against infringement

Developer warrants that Developer will not knowingly infringe on the copyright or trade secrets of any third party in performing services under this Agreement. To the extent any material used by Developer contains matter proprietary to a third party, Developer shall obtain a license from the owner permitting the use of such matter and granting Developer the right to sublicense its use. Developer will not knowingly infringe upon any existing patents of third parties in the performance of services required by this Agreement, but Developer MAKES NO WARRANTY OF NONINFRINGEMENT of any United States or foreign patent.

[OPTIONAL—INDEMNIFICATION FOR CLAIMS:] If any third party brings a lawsuit or proceeding against Customer based upon a claim that the Software breaches the third party's patent, copyright, or trade secrets rights, and it is determined that such infringement has occurred, Developer shall hold Customer harmless against any loss, damage, expense, or cost, including reasonable attorney fees, arising from the claim.

This indemnification obligation shall be effective only if:

- the third-party intellectual property rights involved were known to Developer prior to delivery of the Software;
- Customer has made all payments required by this Agreement;
- Customer has given prompt notice of the claim and permitted Developer to defend; and
- the claim does not result from Customer's modification of the Software.

To reduce or mitigate damages, Developer may at its own expense replace the Software with a noninfringing product.

In this clause the developer warrants that it will not knowingly violate the copyright or trade secret rights of any third party. However, no warranty is made as to patent infringement. This is because it is very difficult for a developer to know for sure whether its software might violate a patent, primarily because many pending patent applications are kept secret. Instead, the

developer simply promises that to the best of the developer's knowledge the software will not infringe any existing patent.

Some customers absolutely insist that the developer indemnify (repay) them if a third party brings a lawsuit claiming the software infringed on the party's intellectual property rights. An optional indemnification provision is provided here. It is as narrowly

drafted as possible. The developer must indemnify the customer only if the third-party intellectual property rights involved were known to the developer prior to delivery of the software. This means, for example, the developer won't be liable if a patent issued covering some element contained in the software after the software was delivered.

Alternative 2: “no-knowledge” representation

Developer represents, BUT DOES NOT WARRANT, that to the best of its knowledge the Software delivered to Customer under this Agreement will not infringe any valid and existing intellectual property right of any third party.

Even better for the developer is to provide no warranty of noninfringement at all, and instead extend the “no-knowledge representation” to copyrights and trade secrets as well as patents. Because there are no uniform national rules, it remains far from clear exactly how far copyright protection for software extends. As a result, it can be difficult or impossible to know for sure if custom software violates the copyright in any similar preexisting programs. In fact, there have been many instances in which a developer's employees have included software elements belonging to a prior employer; as a practical matter, it

can be impossible for a developer to know about or prevent this.

Alternative 3: no warranties or representations

THE SOFTWARE FURNISHED UNDER THIS AGREEMENT IS PROVIDED WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS AGAINST INFRINGEMENT, AND DEVELOPER SHALL NOT INDEMNIFY CUSTOMER AGAINST INFRINGEMENT OF ANY PATENTS, COPYRIGHTS, TRADE SECRETS, OR OTHER PROPRIETARY RIGHTS.

Best of all, from the developer's point of view, is a provision in which the developer states that it makes no warranties or representations of any kind regarding intellectual property infringement and will not indemnify the customer if an infringement claim is made (see discussion of indemnification below). This alternative is most logical when the developer retains ownership of the software; it's unlikely a customer who is to obtain ownership of the software would agree to it. If you use this clause, it should be printed in boldfaced, capital letters.

Limits on Developer's Liability

(a) In no event shall Developer be liable to Customer for lost profits of Customer, or special or consequential damages, even if Developer has been advised of the possibility of such damages.

(b) Developer's total liability under this Agreement for damages, costs, and expenses, regardless of cause, shall not exceed the total amount of fees paid to Developer by Customer under this Agreement [OPTIONAL: "or \$[AMOUNT], whichever is greater"].

(c) Developer shall not be liable for any claim or demand made against Customer by any third party except to the extent such claim or demand relates to copyright, trade secret, or other proprietary rights, and then only as provided in the section of this Agreement entitled Intellectual Property Infringement Claims.

(d) Customer shall indemnify Developer against all claims, liabilities, and costs, including reasonable attorney fees, of defending any third-party claim or suit arising out of the use of the Software provided under this Agreement, other than for infringement of intellectual property rights. Developer shall promptly notify Customer in writing of any third-party claim or suit and Customer shall have the right to fully control the defense and any settlement of such claim or suit.

Although it may be rather frightening and depressing to think about, custom software developers face potentially enormous liabilities. Improperly designed or bug-ridden software can cause the customer serious financial losses for which the customer may look to the developer to make good.

EXAMPLE: BioWorkware creates a custom software system designed to automate a biotechnology laboratory. A few weeks after installation, the system crashes. While BioWorkware tries to find out what went wrong, the lab is forced to purchase software from another vendor to get the lab up and running. As a result of all this, the lab is shut down for a week and loses several experiments potentially worth hundreds of thousands of dollars. The lab demands that BioWorkware repay the losses caused by the crash. To pay this amount, BioWorkware would have to liquidate its business. BioWorkware refuses to pay, and the lab sues.

To prevent a nightmarish scenario of this type from driving them out of business, developers are wise to insist on the following provisions limiting their liability to the customer. These provisions are optional, but can be very helpful for developers.

No liability for consequential damages

The consequential damages arising out of even a modest problem can easily bankrupt a developer. At the very least, the developer should insist that the agreement include a provision excluding it from any liability for incidental or consequential damages arising out of the agreement. Such damages include lost profits or other economic damages arising from a malfunction where the developer had reason to know that such losses could occur if the software malfunctioned. This could include, for example, the value of the experiments lost by the biotech lab in the above example.

Limit on developer's liability to customer

Another way to limit the developer's potential liability is to limit the total liability to the customer to a specified amount. A typical liability limit is the amount paid by the customer for the software. This amounts to a money-back guarantee for the customer, while getting the developer off the hook for a potentially much larger liability. If desired, a liability standard not based on the contract price can be set. Obviously, such a provision highly favors the developer.

Developer's liability for third-party claims

The harm caused by malfunctioning software is not necessarily limited to the customer; it may economically or even physically damage third parties as well.

EXAMPLE: SafeSoft writes a custom software package designed to operate a chemical factory. The software crashes and so does the factory, resulting in a chemical spill costing hundreds of thousands of dollars to clean up. Dozens of suits are brought against the chemical company by property owners affected by the spill. The chemical company demands that SafeSoft pay off these claims.

To avoid this type of scenario, the developer may seek to include in the development agreement a clause providing that it is not liable for any claim made against the customer by any third party (other than for intellectual property infringement if it is providing a warranty against infringement).

Indemnification of developer for third-party claims

If third parties are harmed by malfunctioning software, they'll likely sue everyone involved, including the developer. Both the customer and the developer may be liable for the full amount of such claims. Indemnification provisions are used to require one party to pay the other's attorney fees and damages arising from such claims. Indemnification provisions don't affect the third-party claimant. They simply straighten out liability between the developer and the customer.

In the example above, property owners affected by the chemical spill would undoubtedly sue not only the chemical factory, but SafeSoft as well, claiming that it negligently designed the software. A

developer in this situation could find itself faced with defending itself against suits brought by persons and entities it never heard of. Its attorney fees alone could far exceed what it was paid to create the software.

Ideally, the developer would like the customer to agree to indemnify the developer against such third-party claims. Realizing that their software may adversely affect many people they never contracted with, more and more developers are seeking provisions like this which, of course, highly favor the developer. A developer of software that poses an obvious risk of potential financial and/or physical damage to third parties—for example, software implemented in banking hardware or airline radar—should seriously consider seeking such a provision in the development agreement.

Confidentiality

During the term of this Agreement and for ____ [6 months to 5 years] afterward, Developer will use reasonable care to prevent the unauthorized use or dissemination of Customer's confidential information. Reasonable care means at least the same degree of care Developer uses to protect its own confidential information from unauthorized disclosure.

Confidential information is limited to information clearly marked as confidential, or disclosed orally that is treated as confidential when disclosed and summarized and identified as confidential in a writing delivered to Consultant within 15 days of disclosure.

Confidential information does not include information that:

- the Developer knew before Customer disclosed it;
- is or becomes public knowledge through no fault of Consultant;
- Developer obtains from sources other than Customer who owe no duty of confidentiality to Customer; or
- Developer independently develops.

[OPTIONAL—USE WHERE DEVELOPER OWNS SOFTWARE] Customer acknowledges that the Software is Developer's sole and exclusive property. Customer shall treat the Software on a confidential basis and shall not, at any time, disclose the trade secrets embodied in the Software or supporting documentation to any other person, firm, organization, or employee who does not need to obtain access thereto consistent with Customer's rights under this Agreement. Under no circumstances may Customer modify, reverse compile, or reverse assemble the object code contained in the Software. Customer shall devote its reasonable best efforts to ensure that all persons afforded access to the Software and supporting documentation protect Developer's trade secrets against unauthorized use, dissemination, or disclosure.

Since software developers often have access to their customers' valuable confidential information, customers often seek to impose confidentiality restrictions on them. It's not unreasonable for a customer to want you to keep its secrets away from the eyes and ears of competitors. Unfortunately, however, many of these provisions are worded so broadly that they can make it difficult for

you to work for other clients without fear of violating your duty of confidentiality.

If, like most software developers, you make your living by performing similar services for many firms within the computer industry, insist on a confidentiality provision that is reasonable in scope and defines precisely what information you must keep confidential. Such a provision should last for only a limited time—five years at the most.

The confidentiality provision in this agreement prevents the unauthorized disclosure by the developer of any written material of the customer marked confidential or information disclosed orally that the customer later writes down, marks as confidential, and delivers to you within 15 days. This enables the developer to know for sure what material is, and is not, confidential.

The clause also makes clear that the developer does not have any duty to keep as confidential material that does not qualify as a trade secret, is legitimately learned from persons other than the customer, or is independently developed.

Optional provision—Software confidentiality: If the developer retains ownership of the software, it wants to make sure that the customer treats it as confidential. In this event, this optional clause should be included in the development agreement.

Term of Agreement

This Agreement commences on the date it is executed and shall continue until full performance by both parties, or until earlier terminated by one party under the terms of this Agreement.

The term of a software development should run from the date it is executed (signed by both parties) until full performance or earlier termination or cancellation.

Termination of Agreement

Each party shall have the right to terminate this Agreement by written notice to the other if a party has materially breached any obligation herein and such breach remains uncured for a period of 30 days after written notice of such breach is sent to the other party.

If Developer terminates this Agreement because of Customer's default, all of the following shall apply:

- (a) Customer shall immediately cease use of the Software.
- (b) Customer shall, within ten days of such termination, deliver to Developer all copies and portions of the Software and related materials and documentation in its possession furnished by Developer under this Agreement.
- (c) All amounts payable or accrued to Developer under this Agreement shall become immediately due and payable.
- (d) All rights and licenses granted to Customer under this Agreement shall immediately terminate.

[OPTIONAL:] This Agreement may be terminated by Customer for its convenience upon thirty (30) days' prior written notice to Developer. Upon such termination, all amounts owed to Developer under this Agreement for accepted work shall immediately become due and payable and all rights and licenses granted by Developer to Customer under this Agreement shall immediately terminate.

The language governing how the contract may end is very important to both parties. The provision in this agreement is fairly standard. It permits either party to terminate the agreement if the other materially breaches any of its contractual obligations and fails to remedy the breach within 30 days. A “material” breach means a breach that is serious, rather than minor or trivial. Missing a deadline by one day is not “material”; missing it by three months is. For developers, the material breach most likely to result in termination is the customer’s failure to pay the developer in a timely fashion. If the developer terminates the agreement because of the customer’s default (usually because the customer has failed to pay the developer), the customer should return any software or other materials received from the developer and should pay the developer for the work it has done.

Optional provision—early customer termination: The customer may want the option to terminate the agreement at its convenience if it decides for some reason it doesn’t want to go through with the project. In this event, the customer should have to pay the developer all amounts due for accepted work and relinquish any rights in the software.

Taxes

The charges included here do not include taxes. If Developer is required to pay any federal, state, or local sales, use, property, or value added taxes based on the services provided under this Agreement, the taxes shall be separately billed to Customer. Developer shall not pay any interest or penalties incurred due to late payment or nonpayment of such taxes by Customer.

It is customary in custom software development agreements for the customer to pay any required state and local sales, use or property taxes. Whether or not the developer is required to collect sales taxes, the preceding provision should be included in the agreement, making it clear that the customer will have to pay these and similar taxes. States constantly change their sales tax laws and more and more are beginning to look at services as a good source of sales tax revenue. So this provision could come in handy in the future even if the developer doesn’t really need it now.

Developer an Independent Contractor

Developer is an independent contractor, and neither Developer nor Developer's staff is, or shall be deemed, Client's employees. In its capacity as an independent contractor, Developer agrees and represents, and Customer agrees, as follows:

[INCLUDE ALL OF PROVISIONS 23a-j THAT APPLY]

- (a) Developer has the right to perform services for others during the term of this Agreement subject to noncompetition provisions set out in this Agreement, if any.
- (b) Developer has the sole right to control and direct the means, manner, and method by which the services required by this Agreement will be performed.
- (c) Developer has the right to perform the services required by this Agreement at any place or location and at such times as Developer may determine.
- (d) Developer will furnish all equipment and materials used to provide the services required by this Agreement, except to the extent that Consultant's work must be performed on or with Customer's computer or existing software.
- (e) The services required by this Agreement shall be performed by Developer, or Developer's staff, and Customer shall not be required to hire, supervise, or pay any assistants to help Developer.
- (f) Developer is responsible for paying all ordinary and necessary expenses of its staff.
- (g) Neither Developer nor Developer's staff shall receive any training from Customer in the professional skills necessary to perform the services required by this Agreement.
- (h) Neither Developer nor Developer's staff shall be required to devote themselves full time to the performance of the services required by this Agreement.
- (i) Customer shall not provide insurance coverage of any kind for Developer or Developer's staff.
- (j) Customer shall not withhold from Developer's compensation any amount that would normally be withheld from an employee's pay.

One of the most important functions of the agreement is to help establish that the developer is not the customer's employee. A developer will be considered the customer's employee if the customer has the right to control the developer. This will cause problems not only for the customer, but for the developer as well.

A development agreement that indicates a lack of control on the part of the hiring firm will contribute to a finding of independent contractor status. But such an agreement will not be determinative in and of itself. Simply signing a piece of paper will not make a developer an independent contractor. The agreement must reflect reality—that is, the developer must actually not be controlled on the job by the customer.

The language in this clause addresses most of the factors the IRS and other agencies consider in measuring the degree of control of the hiring firm. All provisions show that the hiring firm lacks the right to control the manner and means by which the developer will perform the agreed-upon services.

Include all of provisions a-j that apply. The more that apply, the more likely that the developer will be viewed as an independent contractor.

Optional Provision: Nonsolicitation of Developer's Employees

Customer agrees not to knowingly hire or solicit Developer's employees during performance of this Agreement and for a period of [TIME PERIOD, USUALLY SIX MONTHS TO TWO YEARS] after termination of this Agreement without Developer's written consent.

One of a software developer's fears is that a customer will hire away a star programmer or other key employee and do future work on program enhancements and modifications in-house. This optional provision is designed to prevent this. It may give the developer some peace of mind.

Disputes

[ALTERNATIVE 1—LITIGATION]

If a dispute arises, either party may take the matter to court.

Mediation and Possible Litigation. If a dispute arises, the parties will try in good faith to settle it through mediation conducted by: _____ [NAME] OR a mediator to be mutually selected. [END ALTERNATIVE 1]

[ALTERNATIVE 2—PARTIES EQUALLY SHARE COSTS OF MEDIATOR]

Each party will cooperate fully and fairly with the mediator and will attempt to reach a mutually satisfactory compromise to the dispute. If the dispute is not resolved within 30 days after it is referred to the mediator, either party may take the matter to court. [END ALTERNATIVE 2]

[ALTERNATIVE 3—MEDIATION AND POSSIBLE ARBITRATION]

If a dispute arises, the parties will try in good faith to settle it through mediation conducted by: _____ [NAME] OR a mediator to be mutually selected.

The parties will share the costs of the mediator equally. Each party will cooperate fully and fairly with the mediator and will attempt to reach a mutually satisfactory compromise to the dispute.

If the dispute is not resolved within 30 days after it is referred to the mediator, it will be arbitrated by: _____ [NAME] OR an arbitrator to be mutually selected. Judgment on the arbitration award may be entered in any court that has jurisdiction over the matter. The arbitrator will allocate costs of arbitration, including attorney fees. [END ALTERNATIVE 3]

For an explanation of the dispute provision alternatives, see Chapter 17.

Attorney Fees

If any legal action is necessary to enforce this Agreement, the prevailing party shall be entitled to reasonable attorney fees, costs, and expenses.

If you have to sue the client in court to enforce the agreement, and win, you normally will not be awarded the amount of your attorney fees unless your agreement requires it. Including such an attorney fees provision in the agreement can be in your interest. It can help make filing a lawsuit economically feasible. It will also give the customer a strong incentive to negotiate with you if you have a good case.

Under this provision, if either person has to sue the other in court to enforce the agreement and wins—that is, becomes the

prevailing party—the loser is required to pay the other person's attorney fees and expenses.

General Provisions

(a) **Complete Agreement:** This Agreement together with all exhibits, appendices, or other attachments, which are incorporated herein by reference, is the sole and entire Agreement between the parties. This Agreement supersedes all prior understandings, agreements, and documentation relating to such subject matter. In the event of a conflict between the provisions of the main body of the Agreement and any attached exhibits, appendices, or other materials, the Agreement shall take precedence.

(b) **Modifications to Agreement:** Modifications and amendments to this Agreement, including any exhibit or appendix hereto, shall be enforceable only if they are in writing and are signed by authorized representatives of both parties.

(c) **Applicable law:** This Agreement will be governed by the laws of the State of [LIST APPLICABLE STATE]

(d) **Notices:** All notices and other communications given in connection with this Agreement shall be in writing and shall be deemed given as follows:

- When delivered personally to the recipient's address as appearing in the introductory paragraph to this Agreement;
- Three days after being deposited in the United States mails, postage prepaid to the recipient's address as appearing in the introductory paragraph to this Agreement; or
- When sent by fax or electronic mail. Notice is effective upon receipt provided that a duplicate copy of the notice is promptly given by first-class or certified mail, or the recipient delivers a written confirmation of receipt.

(e) **No Agency:** Nothing contained herein will be construed as creating any agency, partnership, joint venture, or other form of joint enterprise between the parties.

(f) **Assignment:** The rights and obligations under this Agreement are freely assignable by either party. Client shall retain the obligation to pay if the assignee fails to pay as required by this Agreement.

(g) **Successors and Assigns:** This agreement binds and benefits the heirs, successors, and assigns of the parties.

(h) **Severability:** If a court finds any provision of this Agreement invalid or unenforceable, the remainder of this Agreement will be interpreted so as best to carry out the parties' intent.

For an explanation of these provisions, see Chapter 17, “General Provisions.” Review the section, “Additional Contract Provisions,” in Chapter 8.

Signatures

The end of the main body of the agreement should contain spaces for the parties to sign.

Exhibits

Make sure that all exhibits are attached to all copies of the agreement and that the references to the exhibits in the agreement match the actual exhibits. The exhibits to the agreement will include:

- the Functional Specification;
- the Development Plan; and
- the developer's list of Background Technology. 

How to Use the CD-ROM

Installing the Form Files Onto Your Computer	496
Windows 2000, XP, and Vista Users	497
Macintosh Users	497
Using the Word Processing Files to Create Documents.....	497
Using U.S. Copyright Office Forms.....	499
List of Forms Included on the CD-ROM.....	501

Some of the forms mentioned in Web & Software Development are included on a CD-ROM in the back of the book. This CD-ROM, which can be used with Windows computers, installs files that you use with software programs that are already installed on your computer. It is not a standalone software program. Please read this appendix and the README.TXT file included on the CD-ROM for instructions on using the Forms CD.

Note to Mac users: This CD-ROM and its files should also work on Macintosh computers. Please note, however, that Nolo cannot provide technical support for non-Windows users.

How to View the README File

If you do not know how to view the file README.TXT, insert the Forms CD-ROM into your computer's CD-ROM drive and follow these instructions.

- Windows 2000, XP, and Vista: (1) On your PC's desktop, double click the My Computer icon; (2) double click the icon for the CD-ROM drive into which the Forms CD-ROM was inserted; (3) double click the file README.TXT.
- Macintosh: (1) On your Mac desktop, double click the icon for the CD-ROM that you inserted and (2) double click the file README.TXT.

While the README file is open, print it out by using the Print command in the File menu.

Two different kinds of forms are on the CD-ROM:

- Word processing (RTF) forms that you can open, complete, print, and save with your word processing program (see "Using the Word Processing Files to Create Documents," below), and
- Forms from the United States Copyright Office (PDF) that can be viewed only with Adobe Acrobat Reader 4.0 or higher (see "Using U.S. Copyright Office Forms," below). These forms have "fill-in" text fields and can be completed using your computer. You will not, however, be able to save the completed forms with the filled-in data. PDF forms without fill-in text fields must be printed out and filled in by hand or with a typewriter.

See the end of this appendix for a list of forms, their file names, and their file formats.

Installing the Form Files Onto Your Computer

Before you can do anything with the files on the CD-ROM, you need to install them onto your hard disk. In accordance with U.S. copyright laws, remember that copies of the CD-ROM and its files are for your personal use only.

Insert the Forms CD and do the following:

Windows 2000, XP, and Vista Users

Follow the instructions that appear on the screen. (If nothing happens when you insert the Forms CD-ROM, then (1) double click the My Computer icon; (2) double click the icon for the CD-ROM drive into which the Forms CD-ROM was inserted; (3) double click the file WELCOME.EXE.)

By default, all the files are installed to the \Software Development Forms folder in the \Program Files folder of your computer. A folder called “Software Development Forms” is added to the “Programs” folder of the Start menu.

Macintosh Users

- Step 1:** If the “Software Development CD” window is not open, open it by double clicking the “ Software Development CD “ icon.
- Step 2:** Select the “ Software Development Forms” folder icon.
- Step 3:** Drag and drop the folder icon onto the icon of your hard disk.

Using the Word Processing Files to Create Documents

This section concerns the files for forms that can be opened and edited with your word processing program.

All word processing forms come in rich text format. These files have the extension “.RTF.” For example, the Search Request Form discussed in Chapter 3 is on the file .rtf. All forms, their file names, and their

file formats are listed in at the end of this appendix.

RTF files can be read by most recent word processing programs, including all versions of MS Word for Windows and Macintosh, WordPad for Windows, and recent versions of WordPerfect for Windows and Macintosh.

To use a form from the CD to create your documents you must: (1) open a file in your word processor or text editor; (2) edit the form by filling in the required information; (3) print it out; (4) rename and save your revised file.

The following are general instructions. However, each word processor uses different commands to open, format, save, and print documents. Please read your word processor's manual for specific instructions on performing these tasks.

Do not call Nolo's technical support if you have questions on how to use your word processor or your computer.

Step 1: Opening a File

There are three ways to open the word processing files included on the CD-ROM after you have installed them onto your computer.

Windows users can open a file by selecting its “shortcut” as follows: (1) Click the Windows “Start” button; (2) open the “Programs” folder; (3) open the “Software Development” subfolder; (4) open the “RTF” subfolder; (5) click the shortcut to the form you want to work with.

Both Windows and Macintosh users can open a file directly by double clicking on it. Use My Computer or Windows Explorer (Windows 2000, XP, or Vista) or the Finder (Macintosh) to go to the folder you installed or copied the CD-ROM's files to. Then, double click the specific file you want to open.

You can also open a file from within your word processor. To do this, you must first start your word processor. Then, go to the File menu and choose the Open command. This opens a dialog box where you will tell the program (1) the type of file you want to open (*.RTF) and (2) the location and name of the file (you will need to navigate through the directory tree to get to the folder on your hard disk where the CD's files have been installed).

Where Are the Files Installed?

Windows Users: RTF files are installed by default to a folder named \Software Development Forms\RTF in the \Program Files folder of your computer.

Macintosh Users: RTF files are located in the "RTF" folder within the "Software Development Forms" folder.

Step 2: Editing Your Document

Fill in the appropriate information according to the instructions and sample agreements in the book. Underlines are used to indicate where you need to enter your information,

frequently followed by instructions in brackets. Be sure to delete the underlines and instructions from your edited document. You will also want to make sure that any signature lines in your completed documents appear on a page with at least some text from the document itself.

Editing Forms That Have Optional or Alternative Text

Some of the forms have check boxes before text. The check boxes indicate:

- **Optional text**, where you choose whether to include or exclude the given text.
- **Alternative text**, where you select one alternative to include and exclude the other alternatives.

We recommend that instead of marking the check boxes, you do the following:

Optional text

If you don't want to include optional text, just delete it from your document.

If you do want to include optional text, just leave it in your document.

In either case, delete the check box itself as well as the italicized instructions that the text is optional.

Alternative text

First delete all the alternatives that you do not want to include.

Then delete the remaining check boxes, as well as the italicized instructions that you need to select one of the alternatives provided.

Step 3: Printing Out the Document

Use your word processor's or text editor's "Print" command to print out your document.

Step 4: Saving Your Document

After filling in the form, use the "Save As" command to save and rename the file. Because all the files are "read-only," you will not be able to use the "Save" command. This is for your protection. If you save the file without renaming it, the underlines that indicate where you need to enter your information will be lost, and you will not be able to create a new document with this file without recopying the original file from the CD-ROM.

Using U.S. Copyright Office Forms

Electronic copies of useful forms from the U.S. Copyright Office are included on the CD-ROM in Adobe Acrobat PDF format. You must have the Adobe Reader installed on your computer to use these forms. Adobe Reader is available for all types of Windows and Macintosh systems. If you don't already have this software, you can download it for free at www.adobe.com.

All forms, their file names, and their file formats are listed at the end of this appendix. These form files were created by the U.S. Copyright Office, not by Nolo.

Some of these forms have fill-in text fields. To create your document using these

files, you must: (1) open a file; (2) fill in the text fields using either your mouse or the TAB key on your keyboard to navigate from field to field; (3) print it out.

Saving a filled-in form: Newer U.S. Copyright Office forms are enabled with "document rights" that allow you to save your filled-in form to your hard disk. Version 5.1 or later of Adobe Reader is required. When you open these U.S. Copyright Office forms in Reader, you'll see a document rights message box if this feature is available. If you are using an earlier version of Adobe Reader, you will be prompted to download a newer version that will allow you to save. You will not be able to save U.S. Copyright Office forms that do not have document rights enabled, but you can still print out completed versions of these forms.

Forms without fill-in text fields cannot be filled out using your computer. To create your document using these files, you must: (1) open the file; (2) print it out; (3) complete it by hand or typewriter.

Step 1: Opening a Form

PDF files, like the word processing files, can be opened one of three ways.

Windows users can open a file by selecting its "shortcut" as follows: (1) Click the Windows "Start" button; (2) open the "Programs" folder; (3) open the "Software Development Forms" subfolder; (4) open the "PDF" folder; (5) click the shortcut to the form you want to work with.

Both Windows and Macintosh users can open a file directly by double clicking on it. Use My Computer or Windows Explorer (Windows 2000, XP, or Vista) or the Finder (Macintosh) to go to the folder you created and copied the CD-ROM's files to. Then, double click the specific file you want to open.

You can also open a PDF file from within Adobe Reader. To do this, you must first start Reader. Then, go to the File menu and choose the Open command. This opens a dialog box where you will tell the program the location and name of the file (you will need to navigate through the directory tree to get to the folder on your hard disk where the CD's files have been installed).

Where Are the PDF Files Installed?

- **Windows Users:** PDF files are installed by default to a folder named \Software Development Forms\PDF in the \Program Files folder of your computer.
- **Macintosh Users:** PDF files are located in the "PDF" folder within the "Software Development Forms" folder.

Step 2: Filling in a Form

Use your mouse or the TAB key on your keyboard to navigate from field to field within these forms.

If document rights are enabled, you can save your completed form to disk using Adobe Reader 5.1 or higher.

If document rights are not enabled, be sure to have all the information you will need to complete a form on hand, because you will not be able to save a copy of the filled-in form to disk. You can, however, print out a completed version.

NOTE: This step is only applicable to forms that have been created with fill-in text fields. Forms without fill-in fields must be completed by hand or typewriter after you have printed them out.

Step 3: Printing a Form

Choose Print from the Acrobat Reader File menu. This will open the Print dialog box. In the "Print Range" section of the Print dialog box, select the appropriate print range, then click "OK."

List of Forms Included on the CD-ROM

The following files are in rich text format (RTF):

File Name	Form Title
Confidentiality.rtf	Confidentiality Letter
Acknowledge.rtf	Acknowledgment of Obligations
Assignment.rtf	Copyright Assignment
Contractor1.rtf	Independent Contractor Agreement (Favorable to Hiring Firm)
Contractor2.rtf	Independent Contractor Agreement (Favorable to Independent Contractor)
Software.rtf	Custom Software Development Agreement
Website.rtf	Website Development Agreement
Nontechnical.rtf	Employment Agreement for Nontechnical Employees
Technical.rtf	Employment Agreement for Technical Employees
Shrink-Wrap.rtf	Shrink-Wrap End User License Agreement
Click-Wrap.rtf	Click-Wrap End User License Agreement
Negotiated.rtf	Negotiated License Agreement
Linking.rtf	Linking Agreement
NewEmployer.rtf	Letter to New Employer
Multimedia.rtf	Multimedia License Agreement
Release.rtf	Multimedia Publicity/Privacy Release
Nondisclosure.rtf	General Nondisclosure Agreement
Licensee.rtf	Nondisclosure Agreement for Licensee
BetaTester.rtf	Software Beta Tester Nondisclosure Agreement
Visitor.rtf	Visitor Nondisclosure Agreement
Interview.rtf	Interview Nondisclosure Agreement
Search.rtf	Search Request Form
WebLicense.rtf	Website License Agreement
Notice.rtf	Notice of Claimed Copyright Infringement
Counter.rtf	Notice of Claimed Copyright Infringement

The following files are in Adobe Acrobat PDF format (asterisks indicate forms with fill-in fields):

File Name	Form Title
FORMCA.PDF	Form CA*
FORMCON.PDF	Form CON*
FORMDOC.PDF	Copyright Office Document Coversheet*
FORMPAI.PDF	Form PA*
FORMPAS.PDF	Short Form PA*
FORMTXI.PDF	Form TX*
FORMTXS.PDF	Short Form TX*
FORMVAI.PDF	Form VA*
FORMVAS.PDF	Short Form VA*