Practical Guidance

Software Development Agreement Negotiating and Drafting

A Practical Guidance® Practice Note by Sonia Baldia, Kilpatrick Townsend & Stockton LLP

Sonia Baldia
Kilpatrick Townsend & Stockton LLP

A software development agreement is a contract for the design, development, testing, installation, and implementation of new or customized software, including modification of currently existing software (collectively, referred to as "custom software"). Ongoing maintenance for the custom software may also be included in the agreement, depending on the scope of the project.

This practice note addresses key issues in drafting and negotiating an agreement for the development of custom software between a customer and a company engaged in the business of software development ("developer").

In situations where the customer engages an independent contractor or hires an employee for ongoing software development consulting, you should use an independent contractor or employment agreement, as applicable.

In addition, this practice note addresses only software that a customer controls and uses on its systems. This practice note does not address situations when the customer:

- Controls or uses the software on a system hosted by the software developer or any third party
- Is a government entity
- Must comply with industry-specific regulations, laws, or restrictions to use the software
- Uses or accesses the software outside of the United States

The Software Development Lifecycle (SDLC)

Software developers typically determine the parties' operational objectives, roles, and responsibilities over the course of the project using an evaluative roadmap or framework called the Software Development Life Cycle (SDLC). The SDLC sets out the tasks that the parties perform in a series of phases to achieve certain defined milestones during the software development process, as well as the order in which those tasks are performed. You should attach the agreed-upon SDLC Statement of Work ("SOW") as an exhibit to the agreement to more easily locate and refer to during the project and to ensure that the obligations of the parties set out in the SDLC SOW are made subject to the terms of the agreement.

The SDLC generally contains the following development phases:

- Requirements gathering and analysis
- Design
- Build/development
- Testing (also known as Quality Assurance)
- Implementation/deployment
- Maintenance

Requirements Gathering and Analysis

Before designing any software, the developer and customer typically gather and document a set of requirements describing the customer's business objectives and the role the software will play in meeting those objectives.

While customers sometimes develop these business requirements on their own, often each party's senior

employees or project leads meet over a specified period of time to exchange ideas and gather key information, including:

- The scope of the project
- The customer's project budget
- The customer' project schedule
- The features and functionality of the software
- The security requirements for the software
- The intended users of the software
- How the users will use the software
- User interface
- Data flow
- Data storage
- Data inputs and outputs
- Hardware interface
- Software interface
- System interface (either as standalone software or integrated with another system)
- Software and system response times
- Portability of the software
- Data security and privacy considerations

The final result of this analysis is a mutually agreed-upon list of specifications that summarize how the software product as a whole will function to meet the customer's business requirements. These specifications guide the developer in designing the technical and operational functions of the software.

Design

The developer designs the software from the functional specifications created during the requirements gathering and analysis phase. In the design phase, the parties determine the technical details of how the software will perform and create a set of software design documents on which the developer will rely to build the software. The design documents usually describe the:

- Software architecture
- Data flow within the software architecture
- Organization of the software system
- Components that comprise the system
- Behavior of each component
- Communication between the components with each other and with the outside world

Build/Development

In this phase, the developer produces the actual source code for the software based on the detailed design documents. The developer usually determines the programming language it uses for coding the software based on a variety of factors, including the:

- Industry in which the customer operates
- Complexity of the software
- Operating environment of the software product

Testing (Quality Assurance)

After the software is built, the developer performs quality assurance testing on the software. The quality assurance team tests the code against the requirements in a static environment to make sure that the product is functioning in accordance with the specifications agreed on during the requirements gathering and analysis phase. Testing can also reveal any programming bugs or design defects in the software.

Structure this phase as an iterative process, where the developer tracks and reports any deficiencies, bugs, or defects uncovered during quality assurance testing, and then fixes and re-tests the software until it resolves the problems.

Implementation/Deployment

Once the software is tested, the developer deploys the software on the customer's systems in the production environment or in a replica of the production environment created for user acceptance testing. Upon the customer's sign-off, the software is deployed in "go-live." Depending on the customer's business strategy, the developer may deploy the software in stages by releasing it to only a limited number of the target users at a time (Pilot Testing).

During Pilot Testing, end users provide feedback on their experiences using the software. The parties evaluate the feedback and mutually agree on any necessary modifications that the developer should make before re-releasing the software back to the end users. The parties repeat this process for every deployment with the goal of making each new iteration of the software run more effectively than the previous version. After the parties release the final version, the developer will update software that the existing customer base used during Pilot Testing to the latest version.

Maintenance

After the software is deployed in the "go-live" production environment, the developer may provide ongoing bug fixes, improvements, updates and enhanced functionality via patches and new version releases.

SDLC Models

The parties may use different project management practices and methodologies for the software development project. Two common approaches are waterfall and agile, with each having pros and cons that should be considered at the outset of the contracting process as the approach selected by the parties can materially impact the drafting and negotiations of some of the key provisions in the software development agreement, such as acceptance and testing, milestones, pricing, warranties and termination rights, to name a few.

Waterfall Model

This is a linear, sequential approach to the SDLC that divides the software development activity into distinct phases that are completed step by step. Each phase consists of a series of tasks and objectives, and each phase must be completed in entirety before the next phase can begin, similar to the direction of water flows over the edge of a cliff (as shown in Figure 1 below). Each phase is dependent on the output of the preceding phase and cannot be changed after completion without a formal change order. The lack of flexibility for mid-cycle changes can often lead to incremental price adjustments and project delays.

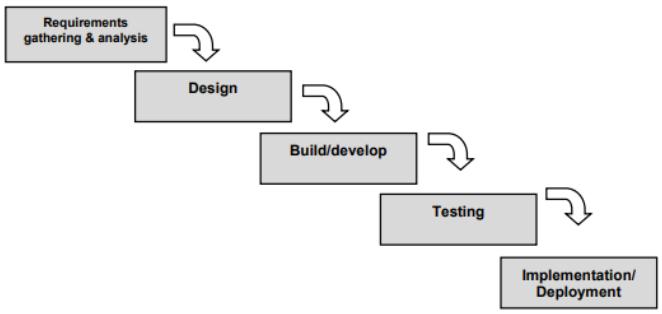

Figure 1: Waterfall model

Generally, this model is suitable for projects with fixed or unchanging requirements that can be precisely defined upfront with minimum probability for changes between the time the software is requisitioned and when it is delivered. While this rigid model provides a clear view of the deliverables, timelines, and accountability for each phase upfront and makes the project easily manageable, it is however not suitable if project requirements are uncertain or frequently changing, or if they require creativity during the development cycle. Typically, the customer has little interaction with the developer during the process so problem identification and resolution does not typically occur until the software is completed.

Agile Model

This approach to project management for the SDLC is centered around incremental and iterative development cycles to complete the project, in which multiple phases can run in parallel, as reflected in Figure 2 below. The software product is broken into small incremental builds (aka "sprints") and items from a list of product features are prioritized and developed (such a list is referred to as the "product backlog"). The goal of each sprint is to produce a working code, with the complete integrated product delivered at the end of the final sprint. A typical sprint can last for about 2–8 weeks.

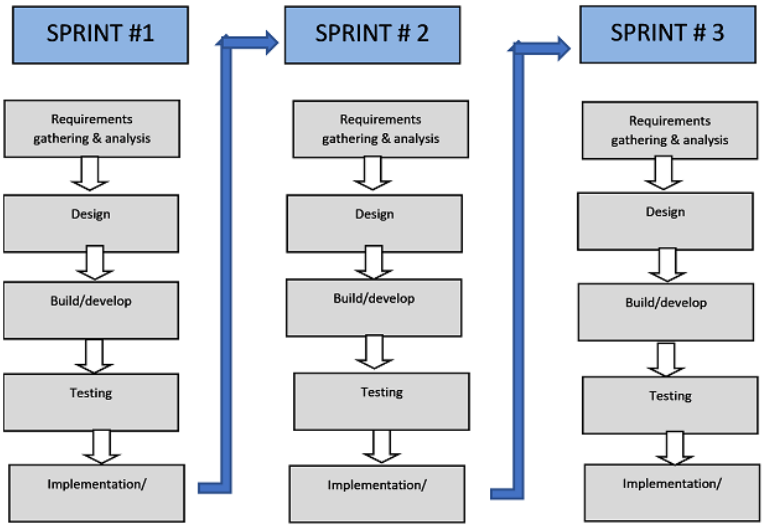

Figure 2: Agile model

This flexible model is increasingly favored over the rigid waterfall model as it allows for closer collaboration between the client and developer to swiftly adjust the code and adapt to changes and operational needs at the end of each iteration. The parties no not rely on creating comprehensive specifications upfront. Instead, the client and developer teams commence work based on a high-level product concept, and define and manage a fluid product backlog as the work progresses. Testing and acceptance occurs in increments at the end of each sprint, unlike in a waterfall project where the entire code is tested at the backend after it is completed.

Using Milestones in Software Development Agreements

An effective software development agreement guides the parties through the stages of the development process by defining key milestones in an effective project plan to reflect the life cycle of the project. Parties use milestones as a tool to:

- Track, monitor and measure the progress of the project
- Minimize the risk of miscommunication by incentivizing each party to provide the other with ongoing feedback
- Identify and resolve problems in the project early on
- Evaluate the potential impacts of a change on the project's schedule and cost

The parties also use milestones to define the crucial project tasks or specific deliverables to be completed during project planning, as well as to specify the activities to be performed and completed by the parties at certain key points throughout the life cycle of the project. The milestones (including deliverables) are typically tied to customer acceptance and payment triggers, and in some instances impose penalties for failure to meet the milestones and/or incentives for early delivery of desired outcomes. Smaller projects usually have fewer milestones (such as the project start date and end date) whereas complex or longer duration projects tend to have multiple milestones that may be broken down into interim and critical deliverables. Generally, milestones are more common in projects using the traditional waterfall approach due to the linear nature of the development cycle and the ability to define detailed specifications at the outset, unlike in agile projects where the development is more fluid with focus on short iterations or sprints requiring collaboration between the customer and developer.

Key Provisions of a Software Development Agreement

Software development agreements typically include the following key provisions. The approach to drafting and negotiating some of these contractual provisions may differ based on the waterfall or agile methodology used by the parties to develop the software.

- Deliverables
- Pricing
- Ancillary Expenses
- Payment
- Change Requests
- Acceptance Testing
- Intellectual Property Rights
- Representations and Warranties
- Indemnification
- Limitation on Liability
- Confidentiality
- Staffing
- Insurance
- Non-solicitation
- Termination

Customer Responsibilities Parties sometimes prefer to draft business terms in a separate Statement of Work (SOW), attached as an exhibit to the agreement in order to locate and refer to it more easily during the project. Examples of key provisions include:

- Description and the scope of the services to be provided
- Detailed project plan
- Project milestones
- Project deliverables
- Testing and acceptance criteria
- Payment schedule and pricing methodology (e.g., fixed fees, T&M)
- Project organization and personnel requirements

Deliverables

Deliverables are a list of tangible items that the parties (typically, developer) deliver during the software development process. The parties typically refer to project milestones, previously agreed on, to create this list. A milestone denotes a key point of accomplishment in the software development lifecycle. A deliverable is a measurable and verifiable item

that a party undertakes to help achieve that milestone. The parties may need to produce one or more deliverables to achieve a specific milestone. You can either incorporate this list within the agreement or attach it as an exhibit.

The developer typically prepares the list of deliverables after the parties agree on the requirements specifications and includes specific due dates for the completion of each deliverable. Deliverables often include:

- Project plan approval
- Functional specifications review
- Detailed design review
- Executable software modules delivered for a specific operation
- Test plan review
- Software deployment
- Iteration/sprint process and duration based on the product backlog (in case of agile projects)

The parties often need to modify the deliverables throughout the software development process. For example, you may need to adjust the developer's estimated deliverable due dates because the customer fails to provide relevant information to the developer in a timely manner. Alternatively, parties sometimes discover that they need to add new deliverables to the list during the project. It is critical to ensure that the Change Request section of the software development agreement (discussed below) allows for efficiently making changes to the deliverables and their due dates. A project plan with detailed milestones is a key staple of a waterfall project, unlike an agile project in which the product concept guides the parties to create and maintain the product backlog and its subsequent iterations.

For the customer: The customer should tie the completion of a set of deliverables or the achievement of a key milestone to a progress payment to the developer to incentivize the developer to work efficiently while producing a successful outcome. Ensure that the interim deliverables are complete and accurate before allowing the developer to commence further work.

For complex projects, consider including a provision allowing a third-party consultant to review and verify the deliverables. In addition, insist that the developer deliverables be provided in short time cycles (e.g., every 5–10 days), as appropriate. With shorter time cycles, the customer can more easily evaluate the project's progression and can provide timely feedback to the developer for any changes or adjustments in real time.

For the developer: If the parties structure the agreement to include progress payments, ensure that the successful delivery of the items described in the deliverables are determined by objective specifications or criteria that the parties have mutually agreed on, not the subjective opinion of the customer.

The developer also should avoid arbitrary deliverable delivery dates, and instead negotiate dates that realistically reflect the time needed for each party to perform its respective obligations to complete every deliverable. In addition, the developer should require the customer to comply with the Acceptance Testing procedure (discussed below) for reviewing each deliverable so that the:

- Customer provides clear acceptance or rejection of each deliverable within a reasonable period of time
- Parties can address issues as they occur instead of later in the project when remedying an issue may be more time-consuming and costly for the developer
- Developer can more quickly invoice and collect a progress payment from the customer

Pricing

The parties usually negotiate pricing for software development services on either a time and materials or fixed price basis. For time and materials arrangements, the developer charges the customer by the hour for its services and for any equipment and materials it purchases specifically to undertake the customer's project.

In fixed price arrangements, the customer pays a defined price to the developer for its services, regardless of how many hours the developer spends on the project.

The parties can also negotiate variations on these pricing methods. For example, the parties could agree that the customer pays the developer on a time and materials basis, but with a fixed cap on the total amount payable.

Generally, fixed pricing is conducive for a waterfall project as the parties are able to precisely scope out the specifications and the level of effort required by the developer at the beginning of the contracting process. In an agile project, however, while the developer may be able to provide some cost estimates for the prioritized items in the "product backlog" or a fixed price "not to exceed" per sprint, the developer has difficulty in accurately estimating the required level of effort for the entire project. Therefore, a time and materials approach (subject to appropriate limits) tends to work better for agile projects. Other hybrid pricing approaches may also be considered depending on the project specifics.

For the customer: Time and materials may be a very inefficient and expensive pricing model for a customer because it does not typically include a fixed limit on how many hours the developer spends on the project. Be aware of the developer attempting to increase its profit margin by

assigning its least experienced employees on the project, while charging the customer its standard hourly rate.

To incentivize the developer to maximize work output, consider capping the total amount payable to the developer. Also consider including "holdback" amount (e.g., 20%–25% of fees) that the customer withholds from accrued payments until the entire project is completed, discount for delayed sprints, and incentive payments that reward the developer for successfully completing a deliverable before its due date. In fixed pricing projects, the customer may also consider negotiating a "pool" of certain number of development hours baked into the project pricing that the customer may use at its discretion.

While fixed price arrangements help provide budgetary certainty for the customer, the developer often charges a much higher internal hourly rate to protect its profit margin in case the developer exceeds the amount of hours it estimates to undertake the project.

In addition, the developer may limit the scope of the project where the customer would need to pay the developer on a time and materials basis for additional deliverables required to finish the project.

For the developer: Time and materials may be the most profitable pricing model for the developer because it is difficult to estimate the amount of time needed to undertake a software development arrangement. If the customer insists on a cap when the developer charges a time and materials rate, consider negotiating bonus payments for successfully completing deliverables before the applicable due dates to help increase the potential revenue.

Ancillary Expenses

The developer should ensure that the agreement addresses additional fees for any ancillary services that the customer desires. Examples include:

- Out-of-pocket expenses
- Training on how to use the software
- Maintenance and support of the software
- Consulting on projects related to the software development agreement

Out-of-Pocket Expenses

Regardless of whether a time and materials or fixed priced model is used, developers often require reimbursement from the customer for at least some of the developer's reasonable out-of-pocket expenses incurred in connection with the project, including:

- Travel and travel-related expenses
- Media costs
- Communication costs (i.e., telephone, postage)
- Federal and state taxes, duties, levies, and assessments on the product or services delivered

Training

The customer may want the developer to provide training services for the customer and its employees on how to use the software. Address the following issues when negotiating a provision for training services:

- Price the developer charges for training (either as a fixed price or time and materials rate)
- Number of training hours/days to which the developer commits
- Number of the customer's employees the developer trains
- Location of the training
- Qualifications and expertise of the training instructors
- Number of instructors that the developer provides

Maintenance and Support

The parties may negotiate maintenance services that the developer provides for the software. The parties can include the terms of these services in a separate maintenance agreement or in the software development agreement. When drafting maintenance services terms in the software development agreement, consider addressing:

- Price of the maintenance services (either as a fixed price or time and materials rate)
- The developer's obligations for maintenance (e.g., to correct any reproducible bug fixes, or to ensure that the software remains in compliance with the specifications)
- Start date of the maintenance period (e.g., after the expiration of any warranty period set out in the software development agreement)
- Frequency of the maintenance
- Whether the developer provides telephone, email, live chat, or on-site maintenance support

Consulting Services

Depending on the nature of the project, the customer may request that the developer provide consulting services for separate projects (such as adding new features) related to the developed software. While the parties usually negotiate a standalone consulting agreement for these services, they can also include the terms in the software development agreement and structure the agreement in a way that allows the parties to add SOWs after the contract execution.

Payment

While the pricing section specifies how much the customer pays the developer for its services, the payment section defines when and how the customer pays the developer.

For time and materials contracts, the developer typically submits an invoice or itemized statement for the number of hours worked on the project, in addition to any out-of-pocket expenses incurred in connection with the project as discussed below. Barring any payment disputes or objections by the customer, the customer makes payment to the developer within a certain amount of time from the date of the invoice or itemized statement (e.g., 30 days from the date of invoice).

For fixed price arrangements, the customer usually pays the developer in installments that are tied to the completion and acceptance of a deliverable, a set of deliverables, or in regular intervals under a payment schedule. In addition, parties typically negotiate for the customer to pay an initial deposit upon the execution of the agreement, and a final payment or holdback upon project completion (e.g., 25% on execution of the agreement, and 25% on deployment of the software).

For the customer: Periodic payments throughout the term of the agreement may motivate the developer to complete the project on schedule and should be tied to the successful completion of specific deliverables or milestones. For fixed price arrangements, try to limit the amount of the initial deposit (usually 5–10% of the overall price), and backload the amount of the final payment that the customer remits on deployment of the software (35–50% of the overall price).

For the developer: Because software development projects can be lengthy, waiting for a single lump payment at the end of the project involves significant financial risk for the developer. As a result, you should negotiate periodic payments that the customer makes throughout the term of the agreement (e.g., after the successful completion of each project milestone, or in regular monthly installments). In addition, the developer should negotiate a larger upfront initial payment to minimize risk to its cash flow and working capital needs while undertaking the project (e.g., 25–50% of the overall price).

Change Requests

During the course of the project, the customer may want to make changes to the requirements or specifications in a way that impacts the project's scope, schedule, or cost. You should include a provision governing how the customer can request modifications to the requirements or specifications. This is often known as a formal change order provision and typically requires that:

- A requested change must be described in writing
- The request can only be agreed to by an authorized representative of the other party
- The other party must accept or reject the requested change within a certain period of time

In addition, consider including an escalation clause that would:

- Require designated executives of each party to meet and attempt in good faith to resolve a dispute over any proposed change order
- Set a time limit for resolving the dispute
- Allow the parties to end the agreement in accordance with its termination and wind down provisions if the parties cannot resolve the dispute

For the customer: If the agreement was originally negotiated on a fixed price basis, the developer may attempt to charge the customer for the requested change(s) on a time and materials basis. You should maintain pricing consistency to protect against overcharging and unintended accounting errors.

If the parties cannot reach an agreement on the change request, the customer should be obligated to pay the developer only for any work product that has been completed and accepted by the customer, and delivered to the customer. Include a provision that the customer owns all such work product and any IP rights in the work product, including any works-in-progress.

For the developer: The developer should include a provision that gives them the right to refuse a change request that preemptively limits the amount the developer can charge for the request. To avoid the risk that this provision becomes a backdoor termination right for the customer, consider limiting the situations in which the customer can terminate the agreement if the parties cannot agree on a change request. For example, restrict the customer from terminating the agreement for requested changes:

- To deliverables that the customer has already accepted
- When the estimated price for the changes would exceed a designated amount
- When the changes would materially change the timeline for completing the project

If the parties cannot reach an agreement on the change request, the customer should pay the developer for any work product that has been completed and in progress. Consider retaining ownership and all IP rights in the work product to disincentivize the customer from early termination.

Acceptance Testing

A software development agreement should include an acceptance procedure for the customer to review and approve each deliverable before the developer proceeds to the next deliverable. Typically, the developer submits the necessary documentation (and any other relevant materials)

to the other party to review once the developer determines that it has successfully produced the deliverable. In an agile project, the unfinished or unaccepted work may shift into the next iteration, which can create complexity for warranty and pricing terms tied to each iteration. The agreement should therefore clearly describe how such situations will be handled by the parties.

For the customer: The software development agreement often provides the customer a specific period of time to review and to respond in writing to the submission. Ensure that the provision allows a sufficient amount of time to test the deliverable (usually 10–15 days). Consider including a provision that requires written notice of acceptance to minimize the risk of the developer relying on oral communications by any of the customer's employees who are not authorized to provide approval.

In addition, you should describe the customer's remedies in a situation when the deliverable fails its acceptance test, including:

- Requiring that the developer fix the deliverable within a specific period of time at no additional cost to the customer
- Allowing the customer to terminate the agreement
- Requiring the developer to remit a partial refund or credit against any amounts paid by the customer
- Defining a repeated failure to submit a deliverable as a material breach of the contract, enabling the customer to recover damages from the developer

For the developer: To prevent delays and to encourage prompt responses, you can include a provision stating that the submission is deemed approved if the reviewing party does not respond within the specified time frame, or when the customer makes the software available for end users. Consider whether the customer should have sole discretion to approve or reject a submission, or if that party can object to the submission only for specific reasons (e.g., the customer can reject a deliverable only if it materially deviates from the requirements specifications).

Include only objective standards for acceptance testing. Disputes often arise when the customer rejects a deliverable for a subjective reason, such as a perception of how the deliverable should operate or the customer's internal expectation of functionality. Instead, require that the customer perform acceptance tests using the requirements specifications or detailed design documents.

Ensure that the customer cannot reject any deliverable that failed due to any third-party software or hardware unless the developer recommended or required that the customer use that software or hardware.

You should also require that the customer:

- Begin acceptance testing immediately upon the developer's submission of the deliverable
- Provide notice of its determination immediately to the developer
- Include a detailed explanation of its rejection in sufficient detail to enable the developer to recreate and to verify the noncompliance

In the event a deliverable has failed acceptance testing, limit the remedies available to the customer by:

- Allowing the developer to fix and to re-submit a deliverable (usually three attempts)
- Allow the customer to terminate the contract if the developer cannot successfully produce a deliverable, and
- Prohibit the customer from recovering monetary damages aside from a partial refund of any fees paid up to the date of termination

Intellectual Property Rights

The agreement must address which party owns the software and all intellectual property (IP) rights, including:

- Right, title and interest in the software
- Software source and object code, all related documentation, and manuals for the software (including all IP rights in these items)
- Developer's work product created during the project (e.g., scripts, product concept, product backlog, internal software, and data used for testing)
- Developer's own pre-existing work product used in the development of the software
- All related copyrights, patents, designs, trademarks, and trade secrets

The types of software ownership models that the parties may negotiate depending on the project specifics include:

- **Customer Ownership.** The customer solely owns the software created by the developer under the agreement, including all IP rights in and to the software.
- **Non-Exclusive License to the Developer.** The customer solely owns the software, but grants the developer a non-exclusive license to use it. Customers sometimes limit the scope of the license so that the developer can only use the software for specific purposes or with third parties that are not competitors of the customer. As the license is non-exclusive, the customer is free to license the software to any third parties.
- **Exclusive License to the Customer.** The developer owns the software and grants the customer an exclusive

license to use it. The developer cannot use or license the software to any third parties.

- **Non-Exclusive License to the Customer.** The developer owns the software and grants the customer a non-exclusive license to the software. As the license is non-exclusive, the developer is free to license the software to any third parties. The developer is restricted from using the feedback and proprietary ideas contributed by the customer.
- **Joint Ownership.** Under this arrangement, both parties own the software and can license it to third parties, subject to any contractual restrictions agreed by the customer and developer.

IP rights give the owner full control over use of the software and any future revenue that it may exploit from this use. Copyright in the software is important especially to the party that desires to make changes to, create derivative works from, adapt, reproduce, and distribute the software.

Because the developer is the creator/author of the software, copyright law automatically grants rights in the software to the developer, even though the software was developed for and paid by the customer. To transfer rights in the copyright to the customer, the software development agreement should include an assignment provision. The agreement should also detail whether the non-owning party will be granted a license to use the software.

For the customer: To transfer or assign ownership of the software effectively, ensure the provision refers to:

- A present grant of assignment
- The software, including both source code and object code
- All associated documentation developed under the agreement
- All ideas, processes, and other know-how developed in connection with the project
- All IP rights in and to the software and documentation

While software agreements frequently include a standard declaration that the developed software is a "work made for hire," it may not actually qualify as such. Thus, to ensure proper transfer to the customer, also include a statement that, to the extent the software is not a "work made for hire," the developer hereby assigns all ownership rights to the customer. For a more detailed discussion, see Works Made for Hire.

The customer also should require the developer to:

- Execute any and all necessary agreements, documents or instruments to perfect the customer's IP rights in the software
- License any of the developer's pre-existing work product, to the extent necessary to be able to use or to modify the software
- In the event the software incorporates any third-party IP Rights, acquire any and all necessary rights or licenses on behalf of the customer

For the developer: Developers usually rely on their own pre-existing, proprietary techniques, know-how, methodologies, utilities, processes, algorithms, and tools to develop software for multiple customers. As a result, you should ensure that the developer retains ownership of all such pre-existing work product, and all IP rights in such work product.

In addition, you should clarify that the developer can use its pre-existing work product in the development of other software for future clients, free and clear of any ownership claims, liens, or approval rights of the customer. If any pre-existing work product is incorporated in the software, you should grant a limited license to the customer to use that work product solely to use the software, and not to use or otherwise exploit it on a standalone basis.

You also should include a clause stating that the developer is not obligated to transfer ownership of the software (or any related IP rights) until the customer has fully paid for the work performed under the agreement.

Representations and Warranties

Parties to a software development agreement typically tailor this provision to address issues that may impair either party's ability to perform its obligations or imperil the customer's ability to use the software freely.

Standard Representations and Warranties.

Examples of general representations and warranties in a software development agreement include:

- The developer has or will have and maintain sufficient resources, facilities, capacity, and personnel to assure that all work will be provided in a timely and workmanlike manner.
- There are no commitments, obligations, or agreements with any third party that would conflict with either party's obligations under the software development agreement, or otherwise restrict a party from entering into the software development agreement.
- During the term of the software development agreement, neither party will enter into any commitment, obligation or agreement that conflicts with its duties under the software development agreement.
- Each party has obtained all licenses and permits required to perform its obligations under the software development agreement.

- Each party will comply with all applicable laws, rules, or regulations during the term of the software development agreement.

Warranty of Software Performance

Many developers of off-the-shelf software sold to a mass market provide the software "as-is." However, customers who contract for custom software development often aggressively negotiate for a warranty that the software will work as it is intended once it is deployed on the customer's systems. To the extent that developers offer a warranty, they use only objective standards to measure the software's performance (e.g., software specifications). Developers typically do not offer performance warranties that use subjective standards to measure the operational effectiveness of the software (e.g., the software is operating to the reasonable satisfaction of the customer). A performance warranty tied to the software specifications is rather typical in a waterfall project but it can be a "mis-fit" in an agile project where the specifications are not pre-determined. In such projects, the performance warranty can be tied to the working code from each iteration meeting the specifications for that iteration, and to the "integrated" working code at the end of the project meeting the collective specifications from each iteration.

For the customer: Ensure that the agreement includes a warranty that the software performs "substantially" in conformance with the functional specifications. This warranty should commence on the final deployment of the software and last for a specific period of time (usually between 30–90 days, factoring in any seasonality or other variations in demand/use). In addition, consider including a warranty that the software will be free of defects (i.e., free from interruption, programming errors, faults, failures, viruses, and bugs).

For the developer: You should limit the warranty to a "material" conformance with the specifications so that the developer is responsible only for fixing significant errors/bugs that significantly impair the customer's ability to use the software.

The developer should also avoid warranting that the software will operate uninterrupted, or be free of defects. While developers are often diligent in checking for defects when building and testing the software, providing a warranty can substantially burden any developer who must audit millions of lines of software code to verify compliance.

In addition, any performance warranty offered by the developer should not cover:

- Unauthorized modifications of the software by the customer or any third party of customer
- Use of the software with third-party software or hardware, or in an environment, or in a manner not originally contemplated by the parties
- Errors or misuse of the software by the customer, its employees, or agents

Warranty Against IP Infringement and Open Source

In the process of developing the software, the developer may purposely or inadvertently incorporate open source code or the IP of a third party without that party's authorization. Because a customer risks being prohibited from using the software if a third party successfully claims an infringement of its IP rights, the customer should require the developer to warrant that the software does not (or will not) infringe or violate any third-party IP rights and that it will not use unapproved open source code in the project.

Indemnification

Software developers often provide indemnification for:

- Personal injury or property damage caused by the developer's personnel or agents, in situations where the developer is performing on-site work
- Any claim that the software infringes on a third party's IP rights
- Breach of specific representations and warranties
- Gross negligence or willful misconduct of the developer's employees or agents

Ensure that the indemnification provision also addresses the mechanics of providing indemnity to the other party, including:

- The indemnified party's obligation to notify the indemnifying party promptly of the pending or threatened action
- Requiring the indemnified party to provide cooperation and technical assistance to the indemnifying party
- Selection of counsel and control of defense of the case
- Whether the indemnifying party may settle the case unilaterally, or only with the indemnified party's approval
- Monetary caps on indemnity (including attorneys' fees), if any

When negotiating an indemnity provision, developers should exclude claims made due (in whole or in part) to the customer's own unauthorized acts. Examples include:

- Modifications to the software without the developer's prior written approval
- Failure to install updates or upgrades to the software that would have avoided the infringement
- Suggested changes to the specifications or software that cause the infringement
- Use of the software with any third-party hardware and software not authorized by the developer

- Use of the software for any reason other than intended purpose
- Gross negligence or willful misconduct of the customer's employees or agents

In lieu of paying damages for IP infringement, software developers typically procure for the customer the right to continue using the software, or modify or replace the software so that it is not infringing. Customers should ensure that any replacements or modifications perform substantially the same as the original software.

While developers usually offer only to refund any fees paid by a customer if they cannot replace or modify the software, customers should include language allowing recovery for all losses resulting from any IP infringement claim.

Limitation on Liability

A limitation of liability clause generally limits the:

- Types of damages recoverable by a party (e.g., special damages, consequential damages, or lost profits)
- Amount of damages recoverable by a party (e.g., a fixed amount, a multiple of fees paid, or an amount recoverable only over a specific period of time)

For the customer: Seek a less restrictive limitation of liability provision because you face less exposure to risk through you own conduct, and more exposure to risk through the conduct of the developer. You should consider:

- Excluding indemnity damages and costs from any limitations on liability
- Avoiding caps on liability for actions resulting in losses that are hard to quantify (e.g., breach of confidentiality, misuse of IP, gross negligence or willful misconduct, or violations of applicable laws)
- Allowing liability caps only for damages that exceed the other party's insurance coverage

For the developer: Seek a more restrictive limitation of liability provision because you face more exposure to risk through you own conduct, and less exposure to risk through the conduct of the customer. As a result, you should consider:

- Capping potential liability to a fixed amount, or a certain amount of revenue generated over a fixed period of time (e.g., fees paid over the past 12 months, or fees paid under the SOW giving rise to the claim)
- Setting a statute of limitations on recoverable damages (e.g., prohibit any action brought more than 12 months after the initial event giving rise to the alleged liability)
- Tying any indemnity remedies to the limitation of liability clause to narrow the scope and amount recoverable
- Capping the aggregate amount of attorneys' fees that a party may recover in a dispute
- Including an aggregate cap that limits total amount recoverable from a party over the life of the contract

Confidentiality

During the course of the software development project, each party will most likely exchange, provide, or disclose information to the other party it deems proprietary and confidential, including:

- Trade secrets
- Research and development materials
- Data used in connection with the software
- Work product created by the developer
- The software, including specifications and documentation
- Customer and vendor lists
- Internal personnel, financial, and marketing information
- Manner and method of conducting business
- Strategic, operations and other business plans and forecasts

If the parties have already entered into a separate confidentiality agreement, you should include a brief provision that references the executed confidentiality agreement and incorporates its terms as part of the software development agreement. Otherwise, you should include a confidentiality clause that prohibits either party from using or disclosing the other party's proprietary and confidential information to third parties. For a more detailed discussion on confidentiality agreements and tips on negotiating confidentiality provisions, see Confidentiality Agreements.

Staffing

Because software developers often rely on key employees who have a specialized skillset or expertise, customers typically require developers to commit to assigning specific individuals to the project. This clause should list out the employees' names, titles, roles and contact information.

For complex projects where the parties need to complete many deliverables, the parties may identify the individuals directly responsible for managing the completion of these deliverables. Parties usually create this roster of directly responsible individuals (DRI) as an exhibit to the agreement or the applicable SOW.

You should also include the name and contact information of specific personnel who will act as the official points of contact for each side. The parties can set out a single person who manages communications, or a schedule of individuals who are responsible for responding to specific types of inquiries.

In addition, a software development agreement often includes a decision tree for managing and escalating day-to-day questions, disputes, and unforeseen issues. Alternatively,

each party can designate a project manager to manage the decision-making process.

The customer typically wants continuity of developer resources for the entire project term but dedicated fixed staffing can be challenging or present additional "bench" costs to the customer in agile projects where the workflow and future requirements are uncertain and may be determined on sprint by sprint basis. The additional cost for the dedicated developer resources should be weighed against the need for continuity on a project by project basis.

For the customer: Include language ensuring developer staffing continuity for the entire project and allowing the customer to require the removal and replacement of a particular project member if, for example, that person violates the customer's on-site work policies, poses a security risk, or engages in any other type of negligent behavior.

For the developer: Refrain from committing specific personnel exclusively to a project. In addition, consider limiting the circumstances under which the customer can require the removal and replacement of a particular project member, such as for repeated offenses or constant poor work performance. You should also avoid providing a specific number of employees that will be staffed on the project to maintain flexibility in undertaking multiple projects at the same time.

Insurance

Developers often agree to maintain the following customary business liability insurance for at least the duration of the software development agreement:

- Commercial general liability insurance covering personal injury and property damage caused by the developer during any on-site work
- Automobile insurance for vehicles owned or operated by the developer
- Worker's compensation in an amount required under the laws of the states where the developer provides the services
- Errors and omission insurance
- Excess liability insurance (also known as Umbrella Insurance)
- Cyber liability insurance if the software interacts with or stores any of customer's Personally Identifiable Information (PII), or otherwise be used in mission-critical applications where third-party intrusions or outages could cause significant economic loss

For the customer: This provision should list out the monetary coverage limits of each type of insurance that the developer maintains. Other important requirements to consider include:

- **Survival.** Require that the developer's coverage continue after the expiration or termination of the agreement to cover claims that the customer would otherwise have the right to make.
- **Additional Insured.** Each insurance policy should specifically name the customer, its agents and employees as additional insureds for all claims arising under the agreement.
- **Certificates.** Ensure that the developer provides you with certificates evidencing the coverages and additional insured designations.
- **Insurance Carrier Rating.** Insist that the insurance is from a reputable and highly rated carrier (e.g., A Best rating).
- **Notice of Change.** The developer should notify you in advance of any cancellation, non-renewal, or material modification of any of the coverages (usually 30 days).

For the developer: If you are providing any services at the customer's premises, you should make this provision mutual to cover any damage sustained to your property or injury to your employees.

Non-Solicitation

A software development agreement inherently offers the opportunity for essential personnel of each party to interact and to collaborate closely on a daily basis. As a result, companies typically attempt to lower the risk of losing valuable employees by including a non-solicitation provision. This clause generally prohibits a party from hiring any of the other party's employees working on the project without written consent during the term of the agreement and for a specific period of time after termination (usually 12 months).

Larger companies usually seek to include certain exceptions to this provision to help make compliance practical. Common exceptions include when an employee:

- Responds to a public advertisement or job posting not specifically targeted to the other party's employees
- Was involuntarily terminated by the original employer prior to the solicitation
- Voluntarily terminated their employment not less than a certain number of days prior to the solicitation (usually 90 days, but for specialized employees or key management you should set a period of 180 days)

Termination

The scope of termination rights and consequences of termination are critical in a software development agreement. The developer usually limits the customer's right to terminate for cause only, whereas the customer usually insists on broader termination triggers. The customary termination

for cause enables a party to terminate the agreement or the relevant SOW for the other party's material breach that remains uncured (say, for 30 days) after notice thereof. Upon such circumstance, the non-breaching party may seek to recover incurred damages, subject of course to any limitations of liability agreed by the parties. Other termination triggers include a party's insolvency or change of control.

The customer can usually terminate the agreement for convenience by paying an early termination or exit fee. In T&M projects, however, a termination fee is atypical unless of course the customer has agreed to pay "bench costs" for dedicated staffing. Similarly, in agile projects the customer usually has the flexibility to terminate at the end of each iteration with no termination fee.

The customer may obligate the developer to knowledge transfer or provide termination assistance to successfully transition the work product to the customer or its successor developer. The parties should pay particular attention to the consequences of termination and the surviving terms.

Customer's Responsibilities

To help perform the services in a complete and timely manner, developers usually include a provision detailing the customer's obligations to cooperate with and to assist the developer during the project. Examples include:

- Designating a team project coordinator who is knowledgeable about the project and authorized to make binding decisions for the customer
- Access to the customer's staff, facilities, working space, and equipment
- Access to computers, software, data, and customer information (even if operated by a third party for the benefit of the customer)
- Securing any necessary third-party authorizations to undertake the services
- Backing up all data, files, and information prior to the commencement of any work, and assuming sole responsibility for this content
- Preparing the customer's systems for the implementation and deployment of the software

In situations where the customer uses or allows access to the software outside of the U.S., the developer should include a provision requiring the customer to comply with all applicable U.S. export laws and regulations. In addition, consider including a requirement for the customer to comply with all import/export laws of any foreign jurisdictions when using or allowing access to the software.

Source Code Escrow

Parties to a software development agreement that involves customizing and maintaining the developer's or third-party software usually execute a source code escrow agreement to protect the customer in the event the developer is unable to continue to develop, or later to support, the software. Under this agreement, the developer provides copies of the software source code to a third-party escrow agent as it is being developed. The agreement authorizes the escrow agent to release the source code to the customer on the occurrence of certain triggering events (e.g., the developer going out of business, laying off a significant portion of its staff, or discontinuing support and/or maintenance for the software).

Software Support Services

This ancillary agreement often accompanies a software development agreement, and provides for the ongoing support of the software by the developer for a certain amount of time after project completion. Support can include providing bug fixes, new versions, or upgrades to the software. Alternatively, the developer's support obligations may be included in the software development agreement or an SOW thereto.

Sonia Baldia, Partner, Kilpatrick Townsend & Stockton LLP

Sonia Baldia brings business and technology savvy to her global practice, which encompasses U.S. and international commercial, transactional, and intellectual property (IP) expertise across multiple industries including life sciences, banking and finance, healthcare, energy, information technology (IT), manufacturing, and software. She advises on a wide array of sourcing, technology, and other commercial transactions and helps companies navigate legal issues raised by data, emerging technologies, and digital transformation, both on the buyer and provider side. Sonia routinely advises clients on IP strategy, management, and monetization arrangements, leveraging her technology background and registered patent attorney credentials.

In addition to her U.S. bar admissions, Sonia is also qualified to practice law in India and she leverages that combined experience on behalf of clients in India-related matters.

Prior to rejoining the firm, Sonia was a partner in the Washington, D.C. office of an international law firm where she was part of its technology, IP, international commercial, and India practices. Sonia has also served as a consultant to the U.S. Agency for International Development (USAID) and the U.S. Department of Commerce in Washington, D.C. where she advised foreign governments on IT, telecom and IP-related development projects. She has also served as associate professor of law, teaching courses in IP, technology transfer, and corporate law.

Sonia was ranked in 2022 and prior years by *Chambers USA: America's Leading Lawyers for Business* in Technology & Outsourcing and in 2020, she was recognized for her expertise in outsourcing deals involving India and her broader technology expertise. She has also been consistently recognized by *Legal 500* amongst the leading practitioners in its technology, media, and telecom outsourcing category (2009-2022; and as a "New Generation Partner" in 2020 and a "Leading Lawyer" in 2021-2022). Sonia was recognized in *The Best Lawyers in America*® for Information Technology Law in 2022 and 2023. She is a frequent speaker and writer on digital transformation, global sourcing, IP, and technology topics and she has authored many articles and book chapters.