Portfolio Media. Inc. | 111 West 19th Street, 5th Floor | New York, NY 10011 | www.law360.com
Phone: +1 646 783 7100 | Fax: +1 646 783 7161 | customerservice@law360.com

Agile Software Development Brings New Contracting Issues

Law360, New York (June 27, 2016, 11:10 PM ET) --

Derek J. Schaffner

Creating software using an agile software development ("ASD") methodology is not a new concept, but it is rapidly gaining popularity among software developers based on the notion that ASD yields workable code sooner and in a more efficient manner. However, traditional "waterfall" software development approaches do not easily lend themselves to contracting under an ASD approach. In fact, clients and their legal teams may feel very uncomfortable contracting under an ASD approach that is often light on up-front specifications and relies more on collaboration and trust between client and developer. However, there are contractual mechanisms that clients can implement to reduce the uncertainty under ASD while still reaping the benefits of this collaborative development method. This article explores some of the issues to consider and offers suggestions on the contractual protections a client should strive to include in a software development agreement using ASD.

Waterfall vs. Agile Software Development

Developing software under the traditional waterfall development approach aligns well with traditional contracting techniques. Under the waterfall approach, there is an assumption at the time of contract execution that the development team can define, with some specificity, the ultimate "thing" to be created supported by a detailed project plan and key milestones tied to client acceptance and financial payment triggers. If the assumptions in the project plan are incorrect or need to be modified, the client and developer need to process those changes through the contract's change-management process, which may add incremental costs and time to the software development cycle. These concepts are very easy to memorialize in a development agreement due to the linear nature of a waterfall approach that commences with detailed planning, followed by design, coding, testing (including acceptance), and deployment.

In contrast, software development under an agile methodology is rooted in the ASD manifesto, which states:

> We are uncovering better ways of developing
> software by doing it and helping others do it.
> Through this work we have come to value:
>
> Individuals and interactions over processes and tools

Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.[1]

While the manifesto acknowledges contracts and plans, ASD simply prefers collaboration and real-time response to change over those items. ASD essentially rejects the fundamental waterfall assumption that the "thing" being developed can be defined at the beginning of the project with enough precision and detail to create milestones and tie payment to achieving those milestones. Many developers share this belief — software development is not like a construction project with discrete, measurable guideposts and, as a consequence, most organizations cannot "spec out" a software development project with sufficient detail at the time of contract execution. As a result, there is a higher likelihood of "failed" software development projects under the waterfall approach due to this lack of ability to create very good specifications at the outset and the rigidity of the change management process. Developers in favor of an ASD approach believe that the software development process is more fluid, requiring greater interaction between business and technical teams as the project moves through the development life cycle.

Additionally, ASD is preferred by many software developers because it enables software to be developed in continuous cycles based on short iterations, which developers find more efficient and creative. The steps to developing software under a waterfall approach (planning, design, coding, testing, deployment) are still present in ASD, but the work is divided into smaller iterations (or sprints) of two to four weeks with the goal to produce "workable" code at the end of each iteration. Under an ASD approach, the need to test the entire system is minimized since testing (and acceptance) occurs at each iteration.

Contractual Issues

The waterfall approach to software development prefers "bright line" contractual rules, whereas an ASD approach emphasizes unstructured, continuous interactions between the client and developer based on trust and understanding. So how should parties structure software development agreements to take advantage of the creative nature of ASD while still providing important contractual protections akin to those enjoyed under the waterfall approach?

Pricing

Waterfall development allows the client to contract on a fixed-price basis — the deliverables and scope of work have been thoroughly vetted, thereby allowing the developer to estimate the required level of effort and provide firm pricing. However, fixed-price models do not work well for ASD — the detailed specifications have yet to be refined and, consequently, the developer cannot accurately estimate the level of effort required for the project. The ASD process more naturally fits a time-and-materials ("T&M") model, which understandably makes chief financial officers and chief information officers nervous. After all, ASD requires a leap of faith — specifications are not clearly defined and yet there is a pricing model that motivates the developer to charge as many hours as possible. However, this risk can be minimized if a client is working with a known partner who is motivated to work efficiently to avoid losing existing or future client work.

There are variations of the T&M model that can help control costs, such as capped T&M on an iteration or project basis, a "holdback" for each iteration that accrues but is not paid to the developer until the entire project is complete, and a pool or bucket of development hours paid on a fixed basis that the client can spend as it desires. However, the differentiating factor that makes the T&M model for ASD palatable are easy termination rights, which are discussed in the next section. While the ability to easily exit an ASD project is helpful, there is a risk that a critical project will not be completed or not completed in a timely manner. Client-friendly termination rights also have a short shelf life; once the project is sufficiently far along, the client's desire to complete the project often outweighs the flexibility to easily exit the project. This could create the risk of price gouging unless there is a cap on fees.

Another pricing issue under an ASD approach deals with the lack of milestones. Under the waterfall approach, client acceptance of a contract deliverable at a predetermined milestone customarily triggers payment to the developer. Advocates of the waterfall approach may view the absence of milestones under ASD as a lack of control.

The waterfall approach assumes that the parties have done a thorough job capturing all the good ideas at contract execution to create milestones. By its nature, an ASD approach reduces the pressure to correctly identify all the milestones at the outset. The ASD approach begins with a high-level concept of the product to be developed (the "product vision"). Using the product vision as a guide, the development team then creates a statement of requirements (the "product backlog"), which essentially is a list of prioritized items to be developed during the project. The initial product backlog can be included as a contractual document or developed after contract execution. Nonetheless, the development agreement should specify how the product backlog will be managed during the project, including obligations for the developer to provide cost estimates to develop each item in the product backlog. The parties can then use the prioritized list of items from the product backlog to determine the order in which items will be worked.

Since each iteration involves the same activities followed under the waterfall approach (planning, design, coding, testing and deployment) as well as a "definition of done" for that iteration, the milestone framework for the entire project is no longer necessary — the parties decide on the prioritization of items from the product backlog and then determine what the successful completion of that iteration means. In summary, the ASD approach replaces the need to precisely define all the milestones at the outset of the agreement with a methodology that allows the parties to continually assess project status and determine what should be worked on next.

Termination Rights

Typical waterfall development agreements offer customary termination rights — namely, that the client can only terminate the agreement for cause due to material breach or insolvency, or can terminate for convenience upon meeting the notice requirement and possibly payment of termination charges. However, such restrictions do not mesh well with the nature of an ASD agreement. Given that the goal of each ASD iteration is to produce workable code paid for on a T&M basis, the typical ASD agreement allows the client to terminate at the end of each iteration without the payment of termination charges. Therefore, if the client does not see value in the work product produced during the latest iteration, it can simply walk away. The chance that a project goes dramatically astray during an iteration is minimal because the requirements for that iteration are always defined at the outset of that iteration, which presumably advances the larger goals of the "thing" being built in accordance with the product vision. Note, however, that the ASD approach involves minimal software documentation, so restarting a terminated ASD project at a later date may be more costly since new developers will need to spend

more time diving into the code before work can resume.

It is also inappropriate for a developer to seek termination charges under most ASD scenarios. Given the uncertain nature of the workflow, the developer has little visibility into future requirements and therefore cannot dedicate a pool of fixed resources for an extended period of time. Consequently, there should be minimal "bench" costs upon termination. However, a client should weigh the lack of bench costs against the need for developer personnel continuity. For example, a client may agree to pay more to ensure the availability of a core team of developer personnel if the developer is hired to create a mission-critical application.

It is these easy termination rights that allow the T&M model to work for ASD projects. Those in favor of the ASD approach believe it is better to have controlled and uncapped expenses that produce "working" software at each iteration in lieu of uncontrolled capped expenses that may not produce anything of value.

Project Planning

The first contract deliverable under a waterfall development project is often the project plan. In contrast, an ASD approach works best when the project is structured in smaller, discrete components that can be reprioritized as needed and whose specifications are determined when the particular iteration commences. Consequently, project plans rarely exist under ASD; however, the product vision serves as the guiding principle for the product backlog and all subsequent iterations.

If the parties decide to exclude the initial product backlog as a contractual document under the development agreement, the parties should specify in the agreement how the product backlog will be developed and prioritized. The client should also have the sole and exclusive right to amend and reprioritize the product backlog.

An ASD development agreement should also define the iteration process, including the duration of each iteration (usually not to be extended; instead, unfinished work rolls into the product backlog and is prioritized), the meeting cadence, and the process the parties will use to determine when the work in an iteration is "done."

Change Management

The role of change management under an ASD project is significantly different than under a waterfall project. Please recall that a waterfall approach requires a large amount of precision in the initial project plan, deliverables, and milestones. As a result, any modifications to these contractual items after execution will need to be processed via the formal change-management process that is also memorialized in the development agreement. Conversely, ASD expects change to occur, even scope creep. However, the product backlog and iteration processes under ASD can be more accommodating when changes occur. Nonetheless, relying on the development team to work through these issues before an iteration commences may be an uncomfortable situation for a client who desires more rigidity in the process and predictability in pricing. The ability to quickly terminate the agreement without termination charges should provide some solace for clients who may be concerned about the lack of formal change management.

Developer Personnel

While the iteration approach under ASD provides a client with the ability to easily terminate a project, developers may be less likely to commit developer resources on a long-term basis unless the client makes a financial commitment to use those resources. This could lead to a lack of continuity if developer personnel leave the client's project after the completion of an iteration and move on to projects for other customers. To mitigate this risk and the subsequent lack of project knowledge, the client should name a few key developer personnel whose commitment to the project is memorialized in the developer agreement with assurances from the developer that these resources will not be moved off the project without approval from the client. The development agreement should also contain assurances that the developer will provide all the resources necessary to complete the tasks in a given iteration, thereby removing the potential excuse that an iteration could not be finished due to lack of resources. Finally, the development agreement should specify that all knowledge-transfer activities among developer personnel are not client-chargeable activities.

Warranties

The lack of specifications at the outset of an ASD project presents a warranty challenge. Under a waterfall approach, the developer could easily provide a warranty that the product meets the specifications and will remediate any failure to perform in accordance with those specifications for a certain period of time. Offering a warranty like this is a bit problematic under ASD, but the developer could warrant that the working code produced during each iteration meets the specifications for that iteration. As more working code is built in subsequent iterations, the client should seek (1) a warranty from the developer that the integrated pieces will work together and (2) a warranty that the entire product will perform in accordance with the summation of the specifications from each iteration. The client should also evaluate whether or not there is some seasonality associated with the product and ensure that the warranty sufficiently covers periods of high demand/use. Also, the client should seek to include standard warranties for intellectual property infringement and the unapproved use of viral open source code.

Intellectual Property Rights

An ASD development agreement should clarify which party owns the intellectual property rights in the developed product and whether the other party has a license to use and/or create derivative works of the developed product. If the developer insists on owning the intellectual property rights in the developed product, the client should have an unrestricted license to the developed product and potentially seek restrictions on the developer's use of proprietary ideas contributed by the client. The situation is further complicated if either party brings preexisting code to the project, which may require case-by-case negotiations as such code is introduced in an iteration. Also, the development agreement should clarify that intellectual property rights in the product vision and product backlog belong to the client.

Acceptance and Product Completion

The parties should memorialize in the development agreement that the client has the ability to test the working code developed in each iteration to determine whether the code complies with the specifications developed at the beginning of the iteration. If the code complies with those specifications, the client can then issue formal "acceptance" of the code. Unless there is sufficient time remaining in the current iteration, remediation efforts for code that fails to comply with the specifications will most likely shift to a new iteration under the product backlog due to the fixed length of iterations. Note, however, that "rolling" unfinished work into a subsequent iteration (especially incomplete or

unaccepted deliverables or items to be fixed as part of a warranty) may have an adverse price impact. Consequently, the parties should clarify how such situations will be handled in the development agreement. The parties should also describe in the development agreement the methodology that will be used to determine when the entire project is completed.

Final Thoughts

Agile software development believes that better software can be created with increased collaboration between the client and developer, as well as business and development teams. While this may indeed be true, contracting for products created under an ASD approach requires a leap of faith by clients who are used to the formality and control found in the traditional waterfall approach. An ASD approach also requires more client involvement across the entire project life cycle, in contrast with the waterfall approach under which the client is heavily involved at the beginning of the process to ensure that the project plan, deliverables and milestones are well defined. This model of continuous client involvement is a big shift for clients who may lack the resources, and possibly talent, to make such a commitment.

Until clients are more comfortable with the lack of control and price variability inherent in ASD projects, the waterfall approach may still be better for the development of mission-critical applications. For those organizations, it is paramount to ensure that the "thing" is successfully built and therefore the rigidity and predictability of the waterfall approach may work better at the expense of a more "creative" solution under an ASD approach. However, clients may consider using an ASD approach for the development of nonmission-critical applications to determine whether the trade-off between the lack of control and the promise of increased creativity is a worthwhile investment.

—By Derek J. Schaffner, Mayer Brown LLP

Derek Schaffner is counsel in Mayer Brown's Washington, D.C., office

The opinions expressed are those of the author(s) and do not necessarily reflect the views of the firm, its clients, or Portfolio Media Inc., or any of its or their respective affiliates. This article is for general information purposes and is not intended to be and should not be taken as legal advice.

[1] http://www.agilemanifesto.org/.

All Content © 2003-2016, Portfolio Media, Inc.