NEW DEVELOPMENTS IN AN AGILE WORLD
DRAFTING SOFTWARE DEVELOPMENT AGREEMENTS

BY PAUL H. ARNE

Software development agreements pose challenges to attorneys and clients alike. Having a robust development process is a key factor in the likelihood of success of any software development project. Project failure rates are high. Both of these factors place demands on those who provide or receive development services and those who memorialize the transaction and allocate risk. Companies are increasingly using a new development methodology, broadly known as "Agile." This relatively new way of developing software poses additional challenges to attorneys.

If you are an attorney who drafts, negotiates, or reviews software development agreements, you need to know about Agile. If you haven't already been brought into a development transaction involving Agile software development, you will. The use of Agile software development principles may change the way your software development agreements should be drafted.[1]

Waterfall Approach

Traditional software development is often described as *waterfall development*. The basic components of waterfall development include the following processes, to be completed in order: conception, analysis (including requirements gathering), design, construction, testing, implementation, production use, and maintenance.[2] Upon completion of each step, the process flows to the next one, hence the term *waterfall*. Inherent in this approach is a substantial attempt to determine as fully as possible what will be built before construction commences.

With waterfall development, changes of plans are generally discouraged and should be carefully scoped before implementation. This places a huge emphasis on getting the design process right. However, there are at least two inherent problems. First, it is hard for human beings to describe what they need when they haven't seen it before. Second, there is a lag between the time something is determined to be needed and the time it is delivered. Changing technologies or business circumstances that arise between the completion of specifications and delivery of the completed software may result in changed needs. Agile development seems to prove more successful.

What Is Agile Software Development?

The name *Agile* can be traced back to one event—in February 2001, 17 software developers met at a Utah resort to discuss lightweight development methods. However, programming styles consistent with Agile principles have been around for much longer. Examples include software development methodologies called *Crystal Clear, Extreme Programming* (or *XP*), *Rational Unified Process, Dynamic Systems Development Method* (or *DSDM*), *Scrum, Adaptive Software Development,* and *Feature-Driven Development*. Many of these techniques are now treated as subsets of Agile.

Out of that event came the "Manifesto for Agile Software Development," set forth below.[3]

> 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

Another result of this meeting was a statement of principles, set forth at the end of this article.

Note the distinct lack of the following as objectives of Agile: predictability of timing, predictability of cost, and clear advance determination of what functionality is to be developed.

Generally, Agile methodologies result in the delivery of working software at relatively short intervals—as often as two weeks. These intervals are frequently called *sprints*. Teams of developers must therefore contain all disciplines necessary to deliver working code. Agile teams will normally be staffed with architects/designers, programmers, and testers.

Industry Trends Point to Agile

Another force that moves software development towards Agile methodologies is the changing method of how software is being delivered—specifically, software as a service (SaaS).

Traditionally, software was delivered and operated on computers owned or controlled by the licensee, and significant software upgrades tended to be major events. On the licensor side, software must be tested against multiple hardware configurations. Interfaces may also need to be tested. Maintenance and support personnel must have the capability to support multiple versions of the software simultaneously, because not all customers will migrate to the new version at the same time. On the licensee side, new versions of software can require extensive testing

Paul H. Arne is the chair of the Technology Transactions Group of Morris, Manning & Martin, L.L.P. The author wishes to thank Mike Cottmeyer for his insights into Agile development, especially for large organizations. Mr. Cottmeyer is the CEO of LeadingAgile, LLC, a company devoted to assisting large organizations transition to Agile development methodologies, including training, coaching, and strategic consulting in portfolio management, project management, and transformation. Also, special thanks to Austin B. Mills for his assistance in preparing this article.

before introduction into a production environment, as well as potentially user training. Because of these costs, based on the author's experience, major new versions of software tend to be delivered no more than about twice a year.

Instead, when software functionality is delivered as a service over the Internet, these dynamics can change. First, software has to be tested against only one hardware environment. Second, there is no compelling reason not to introduce new functionality on an incremental basis as it is completed. With incremental change, training can occur more as an ongoing process rather than being driven by major releases. Support and maintenance personnel frequently only have to deal with one version of the software for all customers. As a result, release cycles for new functionality can be greatly shortened, allowing for new releases as often as a few times a month (sometimes even daily).This new and growing model lends itself to a more iterative software development process, favoring the adoption of Agile methodologies.

Software Development Agreements

Development Agreements

Software development agreements serve the same functions as other contracts. Normal contract functions generally include certainty, roles, risk allocation, and dispute resolution.

Software development relationships usually involve, or should involve, significant interactions between the parties. Guidance in the agreement about *how* the parties will interact with each other during the project can be helpful to set expectations and define responsibilities, and thereby reduce uncertainty and risk. The following terms, among others, can be used in software development agreements to guide the parties, catch and address problems early, assign responsibilities, and manage risk.

- *Development of Specifications*. Because software development agreements are routinely entered into before the parties know specifically what is being created, processes are often built into the agreement related to determining requirements, setting time frames, and providing for the review and approval of the specifications.
- *Establishment of Price*. At times, the cost will not be known either, so there may be a need to provide for a mechanism to determine how much the developer will be paid and when.
- *Management and Decision Making*. Contracts for large-scale development projects often define the organizational structures for managing projects as well as identify who makes what decisions during the course of the project.
- *Milestones*. Software development agreements frequently call for the development and identification of various milestones, and the responsibilities of the parties if these development milestones are missed.
- *Progress Reporting*. Catching problems early can be very important to the ultimate success of a project. Provisions related to what is reported and when, as well as the obligations of the parties based on the substance of the reports, are often found in software development agreements.
- *Acceptance*. The acceptance process normally provides for (i) how completed work is reviewed and confirmed to be in order, (ii) what the responsibilities of the parties are if the completed work is found to be inadequate, and (iii) what the responsibilities of the parties are if the completed work complies with the requirements of the contract.
- *Personnel*. Frequently, software development agreements will provide for how and under what circumstances developer personnel may be added to or eliminated from the development team. For example, it is not unusual to have provisions that prevent key personnel from being removed from the project absent unusual circumstances.
- *Changing Plans*. The process of requesting, scoping, and introducing change into a project is normally addressed.

Using Agile Development Features to Manage the Risk

Some features of Agile development are problematic to attorneys who rightfully seek certainty and risk allocation. When faced with the need to prepare a software development agreement—a kind of agreement already known to be fraught with risk—imagine your client not being able to tell you what is to be developed, how much it will cost, or how long it will take. On top of that, the Agile Manifesto favors customer collaboration over contract negotiation. As an attorney, you may get business pushback as a matter of principle.

However, there are some key features of Agile that help reduce the risk of development projects. These features may be useful to consider when drafting a corresponding software development agreement.

Agile Embraces Changes in Scope. This uncertainty, however, suggests that an ongoing process will exist for evaluating and determining the scope. Therefore, consider identifying what that process is and build into the agreement the requirement of significant participation in scope decisions by both parties.

Also, some kind of scope, anticipated resources, number of sprints, and rough time line still should exist at the beginning of a project. Therefore, consider making sure that at least what is known about scope, timing, and cost are made a part of the agreement.

The agreement should also contemplate how changes to scope are handled in terms of development, time, and money.

It is also important for the attorney to put Agile development in context. Admittedly, in Agile projects it may be challenging to state in a contract specifically what is being developed, at what cost, or how long it will take. However, if more than half the time the reasons cited for projects being over time, over budget, or missing functionality relate to the difficulty of determining requirements or changing needs, how much risk are you reducing by insisting on certainty of scope at the beginning of a project? Also, if statistically software development using Agile methodologies is more likely to be successful, as it seems to be, why would an attorney resist it just because it is less "certain"?

Agile Encourages Close Collaboration Between the Business Unit and the Software Developer. This means that there will be one or more business and developer collaboration processes—structured, informal, or both—that are going to be a part of an Agile development project. Consider building at least some of these processes into the agreement.

Agile Emphasizes the Delivery of Working Software at Relatively Short Intervals Throughout the Project. There are at least two important features of the regular delivery of working code. First, as long as a business can keep what has been built even upon an early termination, then the business may have less risk of having to start all over again. Second, the failure to deliver working code regularly can be a key way to determine whether something is going wrong in the development process. If the code doesn't work, doesn't have the agreed features, or is buggy, then maybe rights should be triggered to get out of the relationship.

Consider building into the development agreement both parties' testing and evaluation of software as a part of each sprint. Also, consider whether there should be a right to terminate at the end of a sprint if the code isn't satisfactory or for convenience.

One More Complication

Unfortunately, everyone seems to have their own interpretation of what Agile development really means. Some developers are more pure in their approach to Agile, while others are clearly not. In the author's practice, he has yet to see two Agile development projects that actually used the same development processes. Therefore, it is important to recognize that the Agile methodologies used by one company will not necessarily match another. Due diligence into the specifics of what is done is an important part of getting the contract right.

Conclusion

Agile development can be uncomfortable to attorneys because of a lack of certainty. However, given success rates of Agile projects compared with other methodologies, the use of Agile seems to be growing. Fortunately, some features of traditional development projects (such as controls related to personnel) are still available to assist with helping to improve the chances for success. Coupled with a sensitivity to features of Agile that are also useful in connection with writing a development contract, lawyers are still in a position to provide value by drafting a document that reduces risk and provides a guide to the parties that may increase the chances for success. ◆

Endnotes

1. This article does not create an attorney/client relationship with you and does not provide specific legal advice to you or your company. Certain legal concepts have not been fully developed and certain legal issues have been stated as fact for which arguments can be made to the contrary, due to space constraints. It is provided for educational purposes only.

2. *See* Wikipedia, *Waterfall model*, http://en.wikipedia.org/wiki/Waterfall_model (as of Oct. 12, 2013).

3. Manifesto for Agile Software Development, http://agilemanifesto.org/ (last visited Oct. 12, 2013) (emphasis in original).

PRINCIPLES BEHIND THE AGILE MANIFESTO

We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.