New Developments in an Agile World:
Drafting Software Development Agreements

By: Paul H. Arne[1,2]

A few months before this article was prepared, a group of senior IT professionals from some of the largest companies in Atlanta[3] gathered together over dinner to discuss issues that are common to large IT organizations. This was the first such meeting, with about 20-25 companies represented. The CIOs who organized the meeting hoped that this would be the first of many such events. The first ever discussion topic? Agile software development. This group has now gathered a second time to discuss a different topic. At the end of that discussion, the audience was asked about topics for the third meeting. The overwhelming response was Agile software development, once again. There's a lot of interest in Agile development.

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.

Software Development Generally

Waterfall Approach

Generally

Traditional software development is often described as "waterfall" development. The basic components of waterfall development reflect a sequential approach to development. Generally speaking, they include the following processes, to be completed in order: conception, analysis (including requirements gathering), design, construction, testing, implementation, production use, and maintenance.[4] Each process flows to the next one, hence the term "waterfall." This process was derived from more traditional disciplines for making things, such as manufacturing or construction. Inherent in this approach is a substantial attempt to determine as fully as possible what will be built before construction commences. One can see how important this principle would be when constructing a building.

Under waterfall development, changes of plans are generally discouraged and should be given special and serious review and scoping before the change is actually implemented. This places a huge emphasis on getting the design process right.

---

[1] Paul is the chair of the Technology Transactions Group of Morris, Manning & Martin, L.L.P. 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.

 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.

[3] Most of these companies have annual revenues in excess of $100 million per year.

[4] *See* Wikipedia, *Waterfall model*, http://en.wikipedia.org/wiki/Waterfall_model (as of October 12, 2013, 16:45 EST).

Very thoughtful and useful techniques have been developed to help organizations figure out what software features and functionality should be built, including stakeholder identification, stakeholder interviews, visioning, brainstorming, naturalistic observation, prototyping, focus groups, and storyboarding. However, requirements gathering and waterfall software development have at least two inherent problems.

- It is hard for human beings to describe what they need when they haven't seen it before. Few people could have envisioned an iPod's interface before actually seeing it.

- There is a lag between the time something is determined to be needed and the time it is delivered. Changing technologies or business circumstances arising between the completion of specifications and delivery may result in changed needs.

Why this is important: Success Rates

Success rates for software development are somewhat difficult to come by. However, the most well-known keeper of these kinds of statistics is The Standish Group. On a regular basis, The Standish Group publishes its "Chaos Report," which is a survey of the success, or not, of IT projects.

Historically, these reports are positively gloomy. For example, in 1995, The Standish Group received survey responses from 365 organizations, representing 8,380 projects.[5] Here are the success rates:[6]

Project Resolution	**Percentage**
Type 1: Project success	16.2%
Type 2: Project challenged: Project completed, but it was over budget, over time, or delivered less functionality than originally planned	52.7%
Type 3: Project cancelled	31.1%

These are scary statistics for IT organizations. As can be seen, cancelled projects were about twice the number of successful projects; over half missed deadlines, were over

[5] *See* The Standish Group International, Inc., *The Chaos Report* (1995), http://www.csus.edu/indiv/v/velianitis/161/ChaosReport.pdf.
[6] *Id.* at 3.

budget, or were delivered without requested functionality. While this author has no industry statistics on this issue, anecdotally — after discussions with the IT litigators in his firm — it should be of little surprise that some of the most frequent disputes in technology are those involving software development relationships.

Faced with these statistics, one can also see how important is it to have development agreements that help manage the development process — especially in times of trouble — and manage disputes.

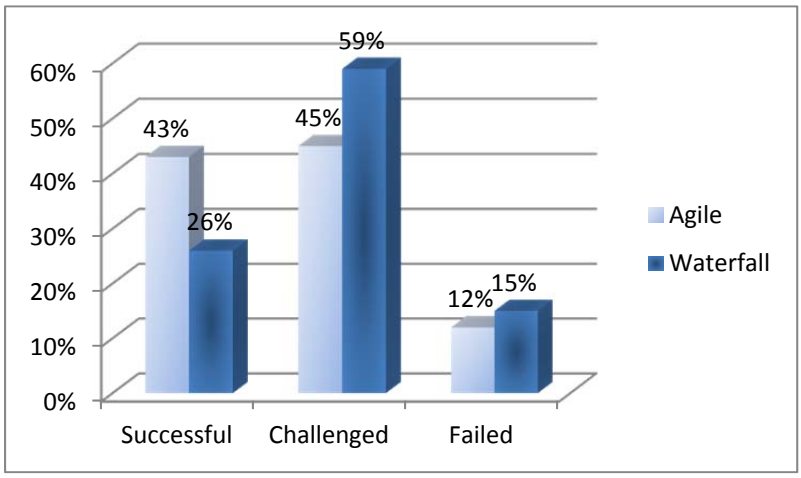
The Standish Group also asked respondents to identify the factors that caused IT projects to be "challenged" (i.e., over time, over budget and/or missing functionality). Of the factors identified, 51.8% of respondents identified factors that relate to the difficulty of determining requirements or changing needs.[7] There seems to be little doubt that human nature and the inexorable pace of technology and business are at odds with some of the fundamental organizing principles of waterfall software development.

The Case for Agile Development

Before getting into what Agile software development is, let's see why it is important.

Success Rates

The Chaos report for 2010[8] showed the following, based on projects from 1994 through 2008:



[7] These response categories were "lack of user input," "incomplete requirements and specifications," "changing requirements and specifications," "unrealistic expectations," "unclear objectives," and "new technology." *Id.* at 5.
[8] *See* The Standish Group International, Inc., *Chaos Summary for 2010* (2010), http://insyght.com.au/special/2010CHAOSSummary.pdf.

One can see from the chart that the use of Agile development methodologies were considerably more successful on average than waterfall methodologies.

From July 22 until November 1, 2011, a different research group[9] surveyed individuals from various competencies within the software development community. 6,042 individuals responded.[10] The median size of respondent's software organizations was 100 personnel.[11] The respondents reported that 80% of their organizations had adopted Agile management techniques in their organization.[12] Sixty percent indicated that over half of their companies' software projects are Agile-based.[13] Comparing Agile-based development projects with prior methodologies used by their companies, between 71% and 84% of respondents indicated that their Agile-based projects (i) increased their ability to manage changing priorities (84%), (ii) improved project visibility (77%), (iii) increased productivity (75%), (iv) improved team morale (72%), and (v) sped time to market (71%).[14]

Not surprisingly, there is a lot of interest in Agile development at the moment.

What Is Agile Software Development?

The name, "Agile," can be traced back to a single event, in February, 2001.[15] While the name grew out of that particular event, programming styles that are consistent with the principles of Agile 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."[16] Many of the listed programming techniques are now treated as subsets of Agile.

Out of that event came the "Manifesto for Agile Software Development," set forth below in its entirety.[17]

---

[9] The survey organization was Analysis.Net Research. The survey was sponsored by VersionOne, which is the provider of an Agile development management tool. Please consider the source of sponsorship when reviewing the statistics; however, some of the survey results are compelling. *See* VersionOne Inc., *State of Agile Survey, 2011, 6th Annual* (2012), http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf.

[10] *Id.*

[11] *Id.* at 1.

[12] *Id.* at 2.

[13] *Id.*

[14] *Id.* at 7.

[15] *See* Wikipedia, *Agile software development*, http://en.wikipedia.org/wiki/Agile_software_development (subheading "Agile Manifesto") (as of October 12, 2013, 17:12 EST). Seventeen software developers met at a resort in Snowbird, Utah to discuss lightweight development methods.

[16] *Id.*

[17] Manifesto for Agile Software Development, http://agilemanifesto.org/ (last visited October 12, 2013).

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.

Another result of this meeting was a statement of principles[18], which is set forth in this article as Exhibit A.

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 results 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 methods of how software is being delivered — specifically, software as a service.

Traditionally, software was delivered and operated on computers owned or controlled by the licensee. This means that delivery of major upgrades to software tend to be major events. Each new upgrade carries a significant cost to both the licensor and licensee. 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 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.

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,

[18] Principles behind the Agile Manifesto, http://agilemanifesto.org/principles.html (last visited October 12, 2013).

there is no compelling reason to not 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.

Because of the above, release cycles for new functionality can be greatly reduced, to as many as a few times a month. New software functionality no longer has to be such a major event. This new and growing model for the delivery of software functionality lends itself to a more iterative software development process. Therefore, certain industry trends favor the adoption of Agile methodologies, at least for some kinds of software.

Software Development Agreements

Purpose of Agreements

It goes without saying that software development agreements serve the same functions as other contracts. Normal contract functions generally include the following:

- *Certainty*. Identifying clearly what the various parties are getting, when, and at what price.
- *Roles*. Establishing which party is responsible for what activities.
- *Allocation of Risk*. If things go badly, then the contract should state the responsibilities of the parties and which party is responsible for the consequences of the problem.
- *Help in times of trouble*. When problems or disputes arise, it can be very helpful to have a process in the contract that enables dispute resolution short of contract termination or litigation.

Development Agreements

Software development relationships usually involve, or should involve, significant interactions between the parties. As a result of these interactions, there are some other objectives in software development agreements that are not as necessary in other contractual relationships. For example, 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. Putting *processes* into these agreements can help manage 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 timeframes, 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 Decisions Making*. Contracts for large scale development projects often define the organizational structures for managing projects as well as identifying 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 if these development milestones are missed, what the responsibilities of the parties are.

- *Progress Reporting*. Catching problems early is often 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.

Describing these processes in software development agreements can help manage the risks.

Using Agile Development Features to Manage the Risk

Some features of Agile development are problematic to attorneys who rightfully seek certainty and allocation of risk. When faced with the need to prepare a software development agreement — a kind of agreement that is already known to be fraught with risk — image your client not being able to tell you (i) what is to be developed, (ii) how much it will cost, and (iii) how long it will take. On top of that, note in the Agile Manifesto a natural bias against contracts generally ("Customer collaboration over contract negotiation"). As an attorney, you may get business pushback as a matter of principle.[19]

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 timeline 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.

It goes without saying that one should contemplate in the agreement 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 less easy to state in a contract specifically what is being developed, at what cost, and how long it will take. However, if over 50% of 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, 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 processes — structured, informal or both

[19] In discussions with Mr. Cottmeyer, who knows some of the people who attended the meeting where the Manifesto for Agile Software Development was developed, he indicated that those in attendance were focusing on "contracts" that occur between IT staff and business units *within* companies and not relationships with outside software developers.

— that are going to be a part of an Agile development project, relating to business and developer collaboration. Therefore, 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. If the developer is to provide *working software* at regular and relatively short intervals during the project, doesn't this reduce the overall risk of 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.

Therefore, consider building into the development agreement the testing and evaluation of software by both parties 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 up to snuff 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 specifically what is done is an important part of getting the contract right.

Conclusions

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.

Exhibit A
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.