Software Development Agreements

A. Waterfall Versus Agile Development Methodologies

Software has traditionally been developed using a waterfall or sequential methodology, where developers complete one phase of the development and then move on to the next phase. Recently, developers have trended toward the “agile” methodology for software development. In the Agile methodology, the delivery approach is rooted in iterative and incremental development. Work is undertaken on an iterative basis in close cooperation with the client. Agile allows for the constant adaptation of the end deliverable. Essentially, developers are working on several work streams simultaneously, allowing for changes to be continually made based on constant client feedback.

The methodology utilized for a development project impacts more than the development process, and both parties must carefully understand how the chosen methodology will impact the project. Under a waterfall methodology, the vendor can provide a clear estimate of the costs for the duration of the project, while under an agile project, the cost estimate at the time of contract signature may change, and there must be a mechanism in place to address price changes. Further, in a waterfall contract the requirements are defined at the start of the contract and amended through change requests, while under the agile methodology the requirements are high-level requirements that can change frequently throughout the project. Delivery dates, including final implementation, are usually determined at time of contract signature under the waterfall methodology, while under the agile methodology a delivery date for a minimally viable product may be established, but other implementation dates will be determined as the project progresses.

When using an agile methodology, it is difficult for a vendor to provide the customer a fixed price, as the development process may be continually changing. Further, a vendor must ensure that any third parties/subcontractors working on an important requirement of the project are also using an agile development methodology to avoid negatively impacting the development process.
The Development Process

Almost all software development projects follow a well-defined project life cycle consisting of 11 phases:

- Development of business requirements (normally supplied by the client)
- Development of business specifications
- Development of functional specifications
- Development of technical specifications
- Development of program specifications
- Unit build and unit/link testing
- System testing
- Client delivery
- Client acceptance
- Post-acceptance implementation
- Post-acceptance support
  - Warranty
  - Maintenance
  - Enhancements

To ensure a successful project, the parties should agree to address the issues raised in each of these phases in significant detail.

In any development project, pricing, functionality, and timing for delivery are the three most important issues. Each one is a function of the other two, and it is important to understand the interrelationship of these issues to ensure a successful project.

1. Pricing

(a) Fixed Pricing

Pricing is the issue which is most negotiated, as most customers seek a fixed-price contract to limit their financial risk. Of course, many vendors are hesitant to agree to a fixed-price contract and assume the financial risk. There are many aspects to fixing the price, and the further into the project the price is fixed, the less risk there is for the vendor. Thus, the question becomes at what time in the development process the price can be fixed to allow each party to feel comfortable with the level of risk they are assuming. To that end, many vendors argue that a fixed-price contract usually will cost the client more than time and materials contract, as the vendor will have to include a significant contingency in its price to protect against losing money on the contract.

If the parties have only agreed on business specifications, there is a very high level of risk, as the technical specifications have yet to be agreed on, which in turn will determine the level of effort. Normally the vendor will only fix the price for the production of the functional specifications and will usually include a large contingency to reflect the incumbent risk.
If the functional specifications have been agreed on, the vendor will usually fix the cost of producing the functional specifications but may include a higher contingency depending on the degree of risk.

If the technical specifications have been completed, the vendor will normally fix the price up to system testing, as the functional specifications allow the vendor to estimate the scope of effort to develop the software with a great deal of certainty.

Under a fixed-price contract, change control must be rigorously enforced for both the vendor and customer to avoid a dispute. Payment is usually made on a percentage-complete basis or on the achievement of key milestones that must be accepted by the customer before payment is made.

(b) Time and Materials
Under a time and materials contract, charges are determined based on the amount of personnel time spent on the project and materials in providing the services in addition to any reimbursable expenses rather than on any results achieved. The client bears the risk of costs that exceed the targeted budget. Payments for time and materials contracts are usually made monthly in arrears based on time sheets submitted by the vendor.

2. Development Risks
Development contracts are fraught with risk from the perspective of both the vendor and the customer. A successful project requires constant diligence on behalf of both parties. One study found that 68% of software projects failed. See http://www.zdnet.com/article/study-68-percent-of-it-projects-fail/ (last visited September 20, 2015).

Set forth below are some of the leading causes of failure. Each party should carefully review each list bi-weekly during the development process to ensure that such issues do not arise.

(a) Customer issues:
- Incomplete specifications
- Customer user makes endless changes to scope/functionality
- Unrealistic time schedules
- Customer provides a detailed RFP and gives the vendor an unrealistic period in which to conduct due diligence and to perform
- Lack of support from the client’s IT staff
- Failure to properly budget for the project
- Inability to make timely decisions
- Lack of proper management by customer to ensure direction and control.

(b) Vendor issues
- Required/identified staff unavailable to staff project
- Incomplete specifications
- Poor quality code and poor quality assurance review
Inexperienced project management
Significant staff turnover
Lack of resources with sufficient skills for new technologies
Personality clashes with client staff

B. Ownership of Developed IP

Ownership of developed intellectual property is often a contentious issue between vendors and customers. Customers almost always argue that they have paid for the development and thus should own the created work. Vendors often seek ownership to re-use the developed intellectual property as part of their existing products or as part of a new product.

To resolve this dichotomy in a logical and thoughtful manner, the parties should discuss how ownership and license rights will work and then draft accordingly. Prior to discussing this issue with the opposing party, each party should meet internally to consider a number of factors that will provide guidance as to the importance of the developed intellectual property. Set forth below are several factors a party should consider when determining the importance of ownership.

For a general discussion, see Porter, Negotiating Rights to a Customer’s Improvements and Modifications, 15 CORP. COUNS. Q. 14 (April 1999). For a discussion of the vendor's point of view, see Gard & Bullock, What’s Wrong With the ‘I Paid For It, I Own It’ Strategy, 23 ACC DOCKET 23 (May 2005).

1. Factors Influencing IP Ownership of Developed Software

In deciding the appropriate position and negotiation strategy, a party should consider the following elements:

- The level of technical development;
- The level of technological advancement;
- The level of difficulty to develop the technology;
- The ease with which the developed technology can be segregated from existing vendor products and solutions;
- The commercial value of the developed intellectual property;
- The existence of a signed-off business case (including committed investment funds) to re-use the developed intellectual property; and
- The value to be gained by the vendor from re-using the developed intellectual property.

In considering the relative importance of ownership, the vendor should distinguish between developed intellectual property that may be re-used for other customers and
developed intellectual property that is created solely for the customer’s environment and is of little or no value to other potential customers of the vendor.

Customers should consider whether the vendor will assign its best technical resources to the project if the customer will own the developed software. Further, will the developed software integrate seamlessly into the vendor’s base code, given that the vendor will most likely not be supporting the code it develops for the customer?

2. Ownership Options

(a) One Party’s Ownership of All Developed IP

The cleanest and simplest resolution to the ownership issue is for one party to own all developed intellectual property. Doing so creates a bright line as to ownership and avoids future disputes over which party owns which aspects of the developed intellectual property. Of course, one party is often unwilling to grant exclusive ownership of the developed intellectual property to the other party, requiring the parties to split the baby in a clear and logical manner.

From the vendor’s perspective, retaining ownership of the developed intellectual property arguably allows the vendor to maintain the integrity of its products and solutions while ensuring their continued improvement and enhancement. Under the vendor’s reasoning, the customer benefits from receiving access to all enhancements and improvements commissioned by other vendor customers as well as reduced maintenance costs from the support of a common product.

Vendors argue that ownership does not benefit the customer, as the customer will likely be unable to independently maintain or support the developed intellectual property and will be forced to pay an additional fee to the vendor or a third party to maintain and support it. Vendors remind customers that the benefits of owning any work product are outweighed by the costs of ownership, including the increased maintenance and support fees arising from exclusive ownership of the intellectual property.

In those situations where the vendor is creating intellectual property that will be part of an existing customer-owned product, the customer will likely demand ownership of the intellectual property and will be unwilling to grant the vendor ownership. In these situations, however, ownership of the developed intellectual property will in all likelihood be of little value to the vendor, and the vendor can concede ownership to the customer.

(b) Ownership Allocated Based on Characteristics of the Software

While most parties focus on an “all or nothing” approach where one party owns all the developed intellectual property, a more productive approach is for both parties to close look at the functionality of the software and determine if it is possible to allocate ownership based on the characteristics of the software and the needs of the parties.

The first step is to review the schematic diagram for the developed software and determine how, if at all, the developed software interacts with the vendor’s solution.
For example, the customer may need to pass on certain ownership rights in the developed software to its own customers and cannot do so if the vendor retains ownership of all developed intellectual property. Notwithstanding the parties’ seemingly conflicting interests, the vendor’s desire to protect the sanctity of its product may not be mutually exclusive with the needs of the customer.

Often the parties negotiating the license may not be technically sophisticated and may not realize that they are negotiating over a particular piece of software that has no value to one party. Thus, it is prudent to review the schematic diagram to ensure that each party understands what the vendor is delivering and the importance of each piece of software, such as an API or interface, to each party. Doing so may provide a common ground for determining ownership, allowing the parties to amicably allocate ownership of discrete portions of the developed software utilizing the schematic diagram. For example, the following language grants the vendor ownership of any developed intellectual property relating to the vendor’s core code language while granting the customer ownership of all other developed intellectual property:

Vendor’s Background IP and any new software developed under this SOW which is part of the Vendor’s Platform that requires re-compilation of the software code of the Vendor’s Platform in order for such new software to function, any improvements to any of the foregoing, and all intellectual property rights therein shall be owned by Vendor.

3. Alternative Ownership Options

Many parties are unable to quickly resolve the ownership issue and enter into protracted negotiations around ownership of the developed intellectual property. To reach a resolution, the vendor should explore with the customer the basis for its concern and then craft a response to address the customer’s concerns. Set forth below are several possibilities for resolving ownership issues.

(a) The Vendor Retains Ownership of All Developed Intellectual Property but Pays the Customer a Royalty or Provides Discounted Services

If the customer believes that because it paid the vendor to develop the intellectual property, it should retain ownership, the vendor can offer to pay the customer a royalty on future licenses of the developed intellectual property to allow the customer to recoup its fees. Model language is set forth below:

In consideration of Customer partially funding the development of the Custom Software, Vendor shall pay Customer a royalty on the future licensing of the Software as set forth in this Section ___. Vendor shall pay to Customer a royalty based on the “Gross License Fee” (“Fee”) of the Custom Software for all third-party licenses of Custom Software by Vendor made within (_______) months from the earlier of [Acceptance] or Vendor first licensing such module to any third party.
While the payment of a royalty is a viable alternative from a business perspective, it may be unwieldy to implement, as the vendor must establish an administrative mechanism to identify, track, and pay royalties to the customer. Further, the customer will most likely insist on a right of audit, which will increase the administrative burden on the vendor. A prudent vendor will seek to limit the customer’s right to audit the vendor to limit disruption to its business and the attendant cost.

Unless the vendor has entered, or envisions entering, into a significant number of royalty-generating contracts, the administrative costs of establishing and administrating the royalty payments may outweigh the benefits to the vendor. Before agreeing to pay a royalty, the vendor should ensure that the business has the capability to track the amounts due and to pay the customer. Also, there is a risk that the customer has an expectation of actually receiving royalties which may not come to fruition, especially if the vendor subsequently decides not to continue to invest in the exploitation of the developed intellectual property. To that end, the vendor should never commit to continue to develop and market the software in question.

From the vendor’s perspective, the vendor should limit its bad debt risk by calculating the royalty on those funds actually received by the vendor, not the funds payable to the vendor by the customer. Payment should be made only when the relicensed intellectual property has been accepted by the subsequent customer. The vendor should not grant multiple customers royalties from enhancements to the same module or software program. That may cause royalties payable by the vendor to potentially exceed its future revenues from relicensing the intellectual property. From the customer’s perspective, royalties should be paid regardless of whether the vendor actually collects payment, as it was the vendor’s choice to contract with the client.

To limit its financial obligations, the vendor should place an absolute cap on the royalties payable to the customer from relicensing the developed intellectual property. The cap can be tied to the amount paid by the customer to the vendor, the amount paid by the customer to the vendor plus a reasonable return, or tied to a set time period. An alternative is to cap the customer’s royalties at a multiple of the fees paid by the customer to the vendor for the developed intellectual property.

From the vendor’s perspective, the vendor should not allow the royalty payments to be unlimited in either amount or the period of time in which the customer is entitled to receive them.

One alternative to paying a royalty is to provide the customer free or discounted services to be applied against future development or maintenance and support. Free or discounted services will likely have less of a financial impact on the vendor and will encourage the customer to purchase additional services from the vendor, potentially expanding the relationship between the vendor and the customer.

Offering the customer a price concession instead of a royalty requires the vendor to assume some risk as under a royalty structure, as the vendor must pay the customer only if the vendor is able to relicense the developed intellectual property. A prudent vendor should not agree to pay a minimum royalty. The benefits of a royalty scheme versus a discount should be balanced against the reduced administrative costs.
associated with a discount in lieu of a royalty as well as potential cost savings if the discount is less than the projected royalty payments.

(b) **The Vendor Retains Ownership of All Developed IP but Agrees to Restrict Future Licenses of the Developed IP**

If the customer is focused on obtaining a competitive advantage in the marketplace, the vendor should suggest a means to prevent the customer’s competitors from using the developed intellectual property for a stated period of time. The vendor can agree to limit future relicensing of the developed IP in a particular geography, industry, or even a particular customer for a limited time period. The following model language reflects this position.

**Ownership of Software Modifications.** In the event Customer desires Vendor to develop a Modification to the Software for Customer’s sole use, Customer must advise Vendor of this restriction prior to development. Upon receipt of such notice, the Modification shall be designated as a “Customer-Specific Modification” and identified as such in the associated Work Order. The parties agree that the Modification shall remain the sole and exclusive property of Vendor; however, Customer shall receive a limited, non-transferable, non-assignable object code license to use such Modification for so long as Customer retains a license to use the Software. Vendor agrees that it shall not license a Customer-Specific Modification to [a third party/named entity or a defined industry] for [a specific time period] after Acceptance.

Any entity/industry limitation should be tightly drawn and limited in duration. The limitation should be related only to the developed intellectual property and not the provision of all services or solutions in the identified geography, industry, or client.

Further, the vendor should seek to retain the right under the underlying agreement to develop functionally equivalent intellectual property provided the vendor does not use the customer’s confidential information. A prudent customer should prohibit the vendor from using the employees and contractors who worked on the developed intellectual property to develop functionally equivalent intellectual property. The customer should also carefully draft the “Residuals” section of the underlying agreement to ensure the customers’ interests are protected and that ownership of any business methodologies created by the vendor in the course of the agreement has been addressed. The following language affirms the vendor’s right to assign the vendor employees who worked on a customer’s project to a competitor’s project.

**Customer recognizes that Vendor’s employees performing the Services under this Agreement and the Statements of Work issued hereunder may perform similar services for others, and this Agreement shall not prevent Vendor from providing services or developing materials that are competitive with those developed or provided hereunder regardless of any similarity to such services or materials, provided, however, that Vendor shall not use or infringe upon any Confidential Information of Customer, any Deliverable, or any of Customer’s other proprietary and intellectual property rights in the performance of such services for others.**
Finally, the customer should also insist on including an audit provision allowing the customer to audit the vendor’s subsequent licensing of the developed intellectual property. Model audit language is set forth below.

Licensor shall keep all usual and proper books and records pertaining to the licensing and use of the Custom Software. During the Term of this Agreement and for three years thereafter, Customer and/or its designated representatives shall have the right to audit (including by inspecting and copying any such books and records) Licensor, in order to verify its compliance with the terms of this Agreement. Customer shall conduct such audits during the Licensor’s normal business hours and in such a manner as not to interfere unreasonably with Licensor’s normal business operations. Customer may conduct such audits from time to time as Customer deems necessary, but shall use any information obtained or observed during the course of the audit solely for the purposes of (i) determining whether the Licensor is making the proper royalties in compliance with the terms of this Agreement and is otherwise in compliance with this Agreement and any applicable laws, and (ii) enforcing its rights under this Agreement and any applicable laws. Except to the extent necessary to enforce its rights, Customer and its representatives will hold all such information in confidence.

(c) Joint Ownership of All Developed IP

A “Joint Work” is “[w]ork prepared by two or more authors with the intention that their contributions be merged into inseparable or interdependent parts of a unitary whole.” 17 U.S.C. § 101. Joint ownership, however, can create more issues than it resolves, as the rights of joint owners vary depending on the type of intellectual property that is jointly owned.

For example, joint owners of a copyrighted work may independently exercise all rights of a copyright owner but are obligated to share profits from licensing the copyright. Joint owners of a patent have no such obligations. Further, the laws governing the joint ownership of intellectual property varies by country, and the effect of one party’s bankruptcy is unclear.

Co-owners of a copyright are treated as tenants in common. Davis v. Blige, 505 F.3d 90 (2d Cir. 2007). Subject to certain limitations, each “owner” is free to license and use the work as it sees fit, as well as transfer its limited rights in the work to a third party. 1 Melville B. Nimmer & David Nimmer, Nimmer on Copyright § 6.11 (Supp. 2006). A co-owner has standing to bring an action for copyright infringement, Isbell Records v. DM Records, 2009 WL 3386546 (5th Cir. Oct. 22, 2009), but cannot grant an exclusive license without the consent of all other co-owners, Sybersound Records v. UAV Corp., 517 F.3d 1137 (9th Cir. 2008) and cannot grant a retroactive license. Davis v. Blige, 505 F.3d 90 (2d Cir. 2007). Each co-owner must also provide an accounting to all other co-owners for any profits. Davis, id. at 98.

Joint ownership allows each party to market the software to whomever it chooses, while at the same time retaining the right to make modifications and enhancements. This alternative may be detrimental to the licensor, because the licensee may license the software to the licensor’s direct competitors. Under joint copyright ownership, however, each owner has a duty to account to the other. 1 Melville B. Nimmer &

David Nimmer, Nimmer on Copyright § 6.12[A] at 36.6 (2005); see, e.g., Oddo v. Ries, 743 F.2d 630 (9th Cir. 1984). While the right to an accounting exists for jointly owned copyrights, there is no right to an accounting for a jointly owned software patent. 35 U.S.C. § 262. This dichotomy could likely lead to confusion, and potentially litigation, if the jointly owned software is protected by both copyright and patent. At the same time, this approach is probably unrealistic as the custom portion most likely is of little value unless it is licensed in conjunction with the rest of the software.

If the resulting work product will contain preexisting intellectual property owned by the vendor and intellectual property owned by the customer, the work product should contain a copyright notice to the following effect:

“This contains material which is the confidential, unpublished property licensed to Customer. Receipt or possession of it does not convey any rights to divulge, reproduce, use, or allow others to use it without the specific written authorization of Customer, and use must conform strictly to the license agreement between user and Customer.

Copyright © 2016 [Licensor Legal Name]. All rights reserved.

Copyright © 2016 [Customer Legal Name]. All rights reserved.”


With joint ownership, all parties with an ownership right must be a party to any suit to enforce an infringement claims. In essence, both parties have a license but they must act jointly to enforce their rights.

(d) Customer Owns All Developed Intellectual Property but Grants the Vendor a Royalty-Free, Perpetual, Non-terminable License to Re-use and Sublicense the Developed Intellectual Property.

Receiving a license allows the vendor to “re-use” sublicense the developed intellectual property, albeit with some restrictions. Under this scenario, the vendor is essentially a reseller. The vendor should seek a royalty-free license so that it may sublicense the developed intellectual property without payment to the customer and seek to retain ownership of all enhancements, improvements, and derivative works made in conjunction with the re-use and sublicensing of the developed intellectual property.

Customer hereby grants Vendor a perpetual, non-transferable, non-exclusive, royalty-free license to use and sublicense the Deliverables with the right to create Derivative works.

See also Checklists for Joint Development Agreements, 90 Forms & Checklists Bi-Monthly 1 (Nov./Dec. 2012)

(e) Customer Owns the Developed Intellectual Property but Issues the Vendor a Covenant Not to Sue

A covenant not to sue offers minimal rights and protection to the vendor. The vendor receives comfort that it may use the developed intellectual property for its internal purposes without fear of legal action, but it may not sublicense the developed intel-
lectual property nor include the developed intellectual property in any of its products. The customer may bring legal action for any other use or further sublicensing. Thus the value to the vendor is minimal. In essence, the value lies in the vendor’s right to use the developed intellectual property for the vendor’s own internal purposes.

C. Ownership of Tools

One aspect of ownership relates to tools such as technical information, plans, designs, templates, processes, methodologies, procedures, reusable software such as source code, object code, routines, and libraries that are commonly used in connection with, and are generic to the development of, a typical computer program. Tools are created in the course of performing work for a customer but have universal application and usually do not constitute confidential customer information. Their universal application to other customers and timesaving mechanism makes them valuable to vendors even though on a stand-alone basis they lack any intrinsic value.

Savvy vendors seek to re-use these tools to avoid having to create new tools for the same tasks in the same way for every customer. If a vendor was forced to re-create a tool for each customer, the tool would be functionally the same but the cost of doing so along with the delay in the project could be great. The wording of some agreements, however, makes the re-use or re-creation of such tools a violation of the customer’s copyright in work products.

Many vendors seek to limit the definition of “work product” to work explicitly developed for the customer during the customer engagement, allowing the vendor to retain ownership of all tools developed in the customer engagement and to use them in future engagements. The ownership of tools should not be an antagonistic issue between the parties unless the tools contain a customer’s confidential information.

Some customers argue that the tools are part of what the customer paid the vendor to create, and thus the customer should receive ownership of the tools. Practically, most tools are of little value to the customer, and the customer usually receives a perpetual, royalty-free license to use the tools in conjunction with the vendor’s deliverables. As such, ceding ownership to the vendor does not hurt the customer and can likely be used as a bargaining chip in its negotiations with the vendor.

See Form B.21 for a model Software Development Agreement.