Agile Contract Options

By Allan Kelly,

Director & Consultant

Agile Contract Options

One question frequent questions asked about Agile methods is: "How do you sign a contract based on Agile working?"

The traditional Waterfall model fits nicely with the way companies buy things: a requirement is drawn up, a supplier quotes a price (based on their interpretation of the requirements and estimates of cost) and everyone signs a legally binding agreement.

There then follows a development period when everyone argues about what is actually in scope, what is out of scope and what constitutes a change request but eventually the work is done, and after some heated debate the customer formally accepts the software and payment is made. The customer gets 200kg of software, the supplier gets their money and everyone is happy - or perhaps not quite everyone.

Of course there is nothing to stop you doing this anyway and then working Agile but it does seem to defeat the whole point of Agile. A change requests is not a big issue for an Agile team, not only do they expect them but lack of change requests would be considered a problem by some. In the extreme Agile teams can start work without any requirements document. Conversely, since deliveries can occur in increments it would seem reasonable to expect early payment.

If both supplier and customer are to obtain the maximum benefit from Agile working then traditional style contracts for work need to be rethought. This is an evolving area and companies searching for new models for Agile contract. Consequently there are opportunities for innovative thinkers to disrupt the status quo.

This article will look at four models available to suppliers and customers. In time new models are likely to appear but right now there are broadly four options.

Debunking fixed contract

The fixed price, fixed duration, fixed scope contract continues to be the standard benchmark for contracts in the IT industry. This contract is based on the idea that an initial project can define the scope of work; from which a supplier can determine what is required - or rather how many people for how long - and thus, calculate a price. Once complete this can be signed in blood and executed as such.

This model is based on the understanding that it is a thing that is being supplied, namely, some quantity of specialist software. Yet the supply of software is less like buying a tailor made suit more like buying financial advice, it is more of a service than a good which is being purchased.

While I have met many people and companies who bid for work on this basis I have yet to meet anyone who has successfully completed such a project without relaxing at least one of the "fixed" parameter. The IT industry has been signing these types of contracts for years and for just as long has been failing to deliver them.

The reasons are not hard to find: once work commences it usually becomes clear that things are missing from the scope, perhaps some items were missed, or perhaps things have changed since the scope was set and, different people interpret the same words differently. Consequently the scope must change, which means duration or staffing must change, which means price and, or, duration must change.

Continued use of this type of contract is touching. It demonstrates the power of faith, positive thinking and hope that next time things will be different.

Perhaps the reason these type of contracts survive is because they can be defended in court. It has been suggested that these contracts are better (for the customer) should a project end in court but actually increase the risk of customers and suppliers going to court.

Not for everyone

Before examining the options it is worth noting that this issue does not effect every company creating software or attempting Agile. Companies which create and sell software products - sometimes called ISVs, Independent Software Vendors, the likes of Adobe and Intuit - only experience this issue on the periphery. Most of their customers only buy products which are finished, and don't get very much direct say in what the product does.

Some corporate IT group who create software avoid the contracting problem too. They buy ready made software (SQL Server) or they create unique software which will be used only by the company itself. Most corporate IT groups at some time contract out work and consequently need to agree contracts. In these cases the IT group is the customer to a third party software supplier.

The companies corporate IT groups sub-contract to - sometimes called External Service Providers, ESPs - are the ones who have most need to contracts which provide for Agile working.

Agile fits easily into the ISV way of working, indeed, many of the Agile practices started life in this segment. Similarly, Agile can fit into the corporate IT world when corporate policies and procedures allow it.

But, ESPs and their customers have problems with the Agile way of working because they need to put a legal contract in place between the two parties. Yet in this problem lies opportunity.

Because customers (corporate IT groups and government departments) expect IT projects to encounter problems contract terms have got stiffer and

financial penalties higher. Only the biggest ESPs can consider bidding on large contracts. Smaller companies cannot afford the penalty clauses and cannot compete with the large players with deep pockets who can afford big penalties.

Changing the contract structure potentially changes the way large IT consumers buy bespoke software and may provide the proverbial win-win scenario. Suppliers who can break away from the fixed price, fixed scope, fixed time contracts stand to gain a competitive advantage over the market leaders. Customers stand to benefit from innovative approaches that provide a better outcome.

Option 1: Hide it

The simplest, least disruptive, way of using Agile within a delivery contract is just to hide it. Don't tell the customer you are working any differently to normal. Estimate and plan the work as you would normally, sign a perfectly normal contract, then use Agile techniques to be better at delivery.

Test driven development, continuous integration, refactoring regular and re-planning will all help you be better at delivering anyway. As far as possible ignore the original plan, it was "wrong" anyway. If possible throw the original plan away.

Trouble is, some customers want to see "progress against plan." You could fake it. I have heard stories of teams who update the plans to make it look like they were following the plan. However this approach isn't entirely truthful, indeed, how can we ever hope to build trust with a customer is we fake part of the process they believe in?

Naturally we don't want to lie to a customer, we want to build trust, and why would they trust us if we were faking following a plan?

Of course, one could argue that customers who wanted to monitor your progress against plan, rather than against actual delivery, are not demonstrating trust either. However it is our job, as contractors, to show them they can trust us.

So to make the "hide it" approach work there needs to be a "don't ask, don't tell" type policy. If you have customer who is prepared to work follow this line, and measure progress against actual deliveries rather than plan then hiding Agile might work. But if you have such an understanding customer then, you probably don't need to hide Agile.

Option 2: No cure, No pay

Adopting a "No cure, No pay" approach requires a certain degree of confidence. The approach is simple: if the customer doesn't like what you deliver there is no fee. However, if they don't pay for what you produce they don't get to keep it either.

While such an approach looks scary it does provide the opportunity to increase the fee. Clients have reduced their risk expose while the contractor has increased theirs. The price for this rebalancing of risk is a higher price.

Such an approach may be intellectually (and perhaps morally) the superior position to adopt but it introduces a new risk. Since the client is less exposed they have less motivation to make the work a success. When decisions get hard to make, compromise are needed, time is scarce, or involvement required the client has no incentive to do what is needed.

When clients have no skin in the game any failure is entirely the failure of the contractor, clients have nothing to loose.

This risk might be offset if suppliers choose to only work with customers who will maintain their commitment. This assume that the contractor feels able to turn down work they judge risky and can correctly assess the commitment level of the client.

To date I have only heard of solo and small suppliers offering this model. The companies with deep pockets are either wary of the risk or don't feel the need to offer this option.

Option 3: Rolling contracts

If we wish to keep customer involved then we need a mechanism to involve them and continually ask them to recommit to the work. This is where rolling contracts have a part to play. Rather than agree a large piece of all-or-nothing work customer and supplier put in place a framework agreement for a series of short development mini-projects, call them episodes or iterations if you prefer.

The contract probably has some overall goal but doesn't contain a shopping list of specific features and functions. The discovery of needs is part of the work itself. One ESP I know of keeps all requirements out of the legal contract. With each iteration something is delivered and, equally important, the understanding of what is needed increases.

With each delivery the customer pays the supplier and has a choice: continue to the next iteration or halt here. The emphasis is put on the supplier to a) deliver something adds value and works, b) demonstrate that there is more worth doing that will add value. If the client cannot see that the value created is greater than the cost they can walk away from the work with that which has been created so far. Equally, if the supplier finds that the client is not co-operating they can walk away too.

In some ways this isn't that different from the traditional practice of paying and recommitting after each distinct phase: requirements, design, development, testing and deployment. The difference is that while in the traditional model a decision to walk away before the end would result in no delivered benefit under this model something is delivered.

On the supplier side there is an clear incentive to demonstrate value through vertical slices of completed functionality - a common Agile practice.

Because the client can see something being created, value being added and a solution coming together they should be enthused to keep working. And because they know they have the option to walk away if they ever loose their commitment they can.

This option moves the legal framework away from the supply of a thing and towards the supply of a service. Clients are contracting for a service and some though needs to be given to the amount of service they are buying. Four man months? 200 velocity points of work?

While this option might sound radical many IT groups and suppliers are already using service contracts in adjacent areas. For example, IT support desks and software maintenance contracts are normally written as service contracts.

Option 4: Money for Nothing, Change for Free

The "Money for Nothing, Change for Free" contract has been documented in detail by Scrum originator Jeff Sutherland. Rather than construct the contract as a framework for mini-projects this approach maintains the big contract - which implicitly suggests some large up front requirements analysis. However two additional clauses are added to the contract.

The first change in the contract exists to facilitate working on the highest priority items first and accommodating new work. Customers agree to meet with suppliers regularly to reprioritise the remaining work. At this time they may add additional work to the backlog on the understanding that in doing so some other work might drop off the end and not get done at all. This increases the incentive to work with and help the supplier and maintenance customer involvement.

Like a rolling contract the customer pays in regular, say monthly, increments in response to delivered working software - which also keeps customers involved, leads to another implicit recommitment and make way for the second change.

The "money for nothing" provision allows the customer at any stage to cancel the remaining work and keep what has been created so far. For this privilege the client pays 20% of the outstanding work.

So if a client cancel's a 12-month $12m project half way through they will pay an additional $1.2m in addition to the $6m paid to date. The client saves the $4.8m they would have needed to spent to see the contract through.

At first sight the supplier looses $4.8m they would have earned. However, the $1.2m they are paid for doing nothing goes a long way to offset this. Assuming they redeploy their staff onto other work quickly much of that $1.2m will be pure profit.

Thus, the mechanisms and incentivise are provided for customers to get involved, get work done early and save money. Similarly suppliers are incentivised to accept change, do good work and collect free money.

At the moment examples of this type of contract are thin on the ground.

Combinations

Given these four options it is easy to see some more alternatives by combining them in different ways. For example, "Change for Free" could be combined with any of the other three options to create a more potent

solution. Similarly "No cure, No pay" could be applied to individual deliveries in option three.

Finances not withstanding options three and four are probably not really very different. Perhaps the main difference is that option three breaks the traditional model because it assumes little is know at the start and asks the customer to repeatedly commit in the positive. Conversely option four works within the existing model, it keeps the assumption of upfront requirements. By providing a get-out clause introduces a rolling contract by the back door..

Which ever way contracts are written for Agile teams there are two essential elements that need to be considered. Firstly, the contracts themselves should embody the iterative nature of Agile working: do a bit, show a bit, do a bit more. This is the theme occurs again and again in Agile: time-boxed iterations, retrospectives, test driven development, etc. etc. It is the PDCA cycle in action.

Second: contracts should incentivise customers and their representatives to maintain involvement with the process for the duration. Study after study has shown continued customer involvement is a key factor in ensuring the success of IT projects. Whether you are working Agile or not you want continued customer involvement.

Last, but not least

Customers who have not been exposed to the IT industry's traditional way of working may find any of these options completely logical. Those who have worked with IT suppliers in the past may find some of these options surprising. Our industry has done customers a disservice by propagating the myth that "if you can write it down, we can build it within a cost and time."

Inevitably some clients will continue to cling to this model. Some suppliers will find good money in taking advantage of these customers, while others will loose money by clinging to a model which doesn't work. Neither option is particularly appealing.

In time, as customers better understand the new ways of approaching IT and the options available everyone stands to benefit. Right now there are opportunities for those who can make an early shift to new contract models. Yes there are risks but there are also rewards.

About the Author

Allan Kelly is London based and works for Software Strategy where he provides training and consulting in Agile practices and bespoke development services. He specialises in working with software product companies, aligning company strategy with products and processes.

In addition to numerous journal articles and conference presentations he is the author of **Business Patterns for Software Developers** (2012) and **Changing Software Development:**

Learning to become Agile (2008). He is also the originator of Retrospective Dialogue Sheets (www.dialoguesheets.com). On Twitter he is @allankellynet and his blog is at http://blog.allankelly.net.

About the Software Strategy

Software Strategy Ltd. is UK based software development consultancy specialising in Agile development processes and practices. Software strategy provides training, consulting and coaching in software development and offers bespoke software development.

Clients range from small web development agencies proving custom solutions to the IT departments of large corporations. Most clients are UK based while others are in Europe and the USA. Clients include Virgin Atlantic Airways, Thompson-Reuters, Iris Legal, Research Instruments, Budget Group Insurance, Fugro and many more.

More about Software Strategy at www.softwarestrategy.co.uk, e-mail contact@softwarestrategy.co.uk or telephone +44 (0) 20 3286 4292.