Is your development contract broken?

Agile development methodologies throw a "spanner in the works" of most traditional development contracts. So much so that if your project is agile, and your contract isn't agile savvy, you could have signed up for a major dispute.

by Stuart van Rij
consultant

It is for this reason that the contracts used for software projects need to reflect the relevant development methodology. To date, most contracts for development work have been prepared on the basis of a waterfall development methodology. This approach broadly caters for a linear plan driven process of requirements gathering, design, build, test and release.

Agile development is fundamentally different and takes a collaborative and iterative approach to the development of software. However, this is an overly simplistic explanation: the closer you get to agile development the more complexity and nuances you discover.

This article focuses on one of the key differences: the extent to which the customer's requirements are fixed or fluid. The differences in this area alone can call for quite different approaches in the contract.

Waterfall projects: fixed requirements

When preparing contracts for waterfall projects lawyers tend to focus on locking down the "iron triangle" of scope, dollars and timetable. In this respect the customer's requirements play a critical role in determining scope and a number of the key features of the contract. For example, the requirements:

- play a governing role in determining the design;
- form the baseline for the acceptance tests;
- feed into some of the central obligations and warranty regimes; and
- once achieved (as evidenced by passing the acceptance tests), trigger significant milestone payments.

Given the integral role of the requirements, they are usually fixed at the start of the project and protected by being given a relatively high priority status in the contract documents. Any changes in those requirements are then managed through the contract change control process.

Agile projects: fluid requirements

Contrast this with the treatment of requirements in an agile project. Rather than locking down the detailed requirements and design at the start of the project, an agile project is more of a voyage of discovery. The requirements are fluid and software is delivered for review after short sharp iterations that each target a subset of the requirements. In many cases each iteration is structured as a mini project in its own right.

One of the key benefits of this approach is that it builds a feedback loop into the project that enables the customer to:

- learn from what has been delivered and then make mid-course corrections to its requirements to better reflect its objectives, changes in its business or any opportunities that arise mid-project;
- prioritise on delivering the most important features first, enabling it to end the project early if it is decides the project is not worth pursuing any further; and
- focus on what can be achieved in collaboration between supplier and customer, given the constraints of high level vision, timeframes and dollars.

But not everything is fluid in an agile project. The measure of progress on an agile project is tested working software and the supplier can be

required to deliver a number of things during each iteration. For, example, the supplier can be required to ensure that each release of software meets certain coding standards, passes certain types of tests and even achieves pre-agreed acceptance criteria. This provides some certainty around the quality of the software and what will happen on an iteration by iteration basis.

- ***Isn't this all a bit too risky?***

The fact that a complete set of requirements isn't nailed down tight at the start of the project can still be a cause for concern. It would seem that loosening the grip on a fixed set of requirements may mean the customer gives up an important means of holding the supplier accountable and getting any comfort as to what must be delivered.

On the other hand, advocates of agile development point out that this desired level of comfort can only give a false sense of security given that the customer's environment and requirements are forever changing. Moreover, the customer may only discover this at the end of the project when it gets to see the software for the first time, and after most of the budget has been spent.

In essence, in an agile project the customer gives up some contractual certainty around the requirements in exchange for the comfort of knowing that the software can evolve throughout the project to better reflect what it discovers it really needs when it actually starts seeing working versions of the product.

- ***The methodology matters***

All of the above serves to illustrate that the methodology matters when you come to build your software development contract. The key thing is to make sure that your contract is closely tailored to the methodology that will actually be used. If it is not, the different contractual requirements around what must be delivered, and how the project will operate, can lead to costly disputes.

Indeed, if the wrong contract has been used there may be a fundamental disconnect between the expectations of the customer and the commercial model of the developer. This tends to end badly. Far better to flush out those gaps at the contracting stage and get a contract framework in place that reflects the reality of the project.

Wigley+Company
PO Box 10842
Level7/107 Customhouse Quay, Wellington
T +64(4) 472 3023 E info@wigleylaw.com
and in Auckland
T +64(9) 307 5957
www.wigleylaw.com