The Curse of the Change Control Mechanism

Contract change management and the limitations of the traditional system for dealing with change are the topics covered by **Susan Atkinson** and **Gabrielle Benefield**. They advocate a better way

We live in fast-moving times. The current pace of innovation is exciting – but relentless, and potentially daunting for anyone trying to keep up. In the technology sector, for example, Apple sold 3 million iPads in the first 80 days following its release in April 2010.[1] In less than a year this has spawned a brand new market for tablet computers, and changed the face of computing. That is great news for consumers and great news for Apple, but it can be challenging for the competition.

Nokia's recent quarter-end results show how quickly a company's market share can evaporate if it doesn't keep up. In just one quarter Nokia's share of the smartphone market dropped from 38% to 31%, because of its failure to produce devices that can compete with Apple's iPhone and smartphones using Google's Android operating system.[2] In the words of Nokia's chief executive *'They changed the game, and today, Apple owns the high-end range'*.[3]

The current pace of innovation doesn't just affect branded products. All aspects of information technology are affected, and all products and services that are underpinned by information technology are affected. And if any of those products or services are procured from third parties, the contracts with the third parties are also affected.

Traditional contract models

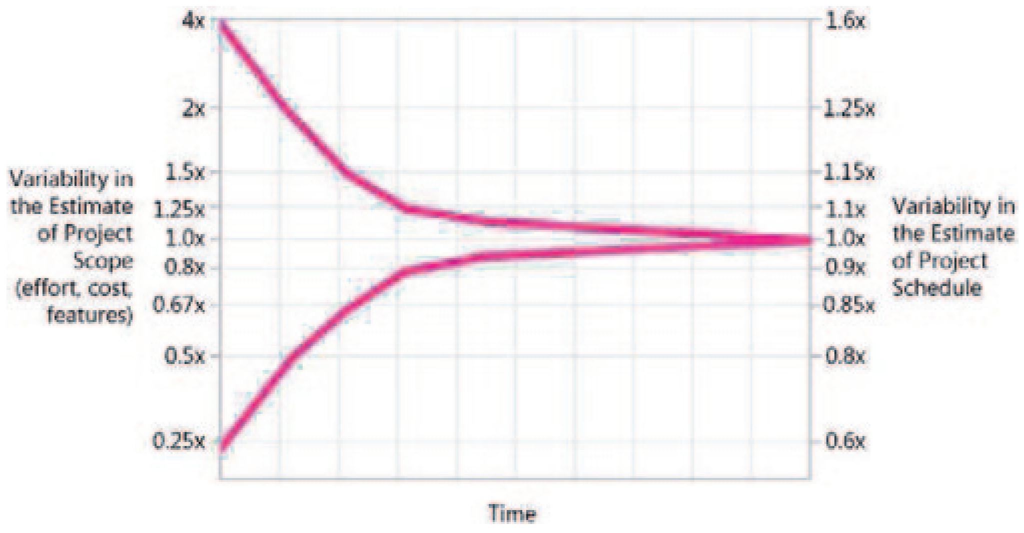
Traditional contract models were designed for commoditised products and services. They are brittle and do not readily embrace change. This is because the products and services are defined upfront, and any change to this definition requires an amendment to the contract, which is usually governed by the change control mechanism. Instead of the change control mechanism embracing change, it is generally regarded as fettering and inhibiting change.

In a traditional contract for the supply of products and/or services, the products/services are described in the contract, the delivered products/services are checked for conformance with the contractual description, and various contractual rights and obligations arise, depending on whether or not the products/services meet the contractual description. This approach is best encapsulated by the expression made famous by the Ronseal advert *'It does exactly what it says on the tin'*.

Such an approach works well for commoditised products and services which can be defined upfront, and there will always be a place for the traditional contract models. However, many organisations are in the business of developing innovative and/or complex products (eg software development) and services (eg the transformation of existing services in business process outsourcing). The issue here is that the products/services cannot be precisely defined upfront. The *'Ronseal label'* is essentially conceptual and only represents an approximate estimate of what is required. There is often great uncertainty surrounding the definition of these innovative and/or complex products and services, and in the course of the supplier trying to deliver what the customer actually wants there will inevitably be change.

The 'Cone of Uncertainty'[4]

Researchers have found that in software development projects estimates are subject to predictable amounts of uncertainty at various stages throughout the project. In this context estimates could outline how much a feature set will cost and how much effort will be required to deliver that feature set, or they could outline how many features can be delivered for a particular amount of effort or schedule. The *'Cone of Uncertainty'* (see figure overleaf) shows how estimates become more accurate as the project progresses.[5]



Software development is a process of gradual refinement. Initially there is a product concept (the vision of the software to be delivered) and that concept is refined, based on the product and project goals. Software development consists of making literally thousands of decisions about all the feature-related issues of the software. Uncertainty in a software estimate results from uncertainty in how the decisions will be resolved. As a greater percentage of those decisions are made, the level of uncertainty should be reduced, and therefore the accuracy of the estimates should be increased.

In other words, the accuracy of a software estimate depends on the level of refinement of the software's definition.[6] The more refined the definition, the more accurate the estimate. The reason why the estimate contains variability is that the software development project itself contains variability. The only way to reduce the variability in the estimate is to reduce the variability in the project.

It is important to appreciate that the Cone of Uncertainty represents the best-case accuracy that it is possible to achieve in estimates at different points in a software development project. This is for two key reasons. First, the Cone of Uncertainty represents the error in estimates created by skilled estimators. Secondly, the Cone of Uncertainty is modelled against well controlled projects. So, if a software development project is not well controlled or the estimators are not skilled, the project will not drive out enough variability to support more accurate estimates, and it is likely that at any point in time the estimates will be less accurate than the Cone of Uncertainty would suggest.

The presence of variability, and therefore uncertainty, is not limited to software development projects. It is also found, although to a lesser degree, in any project for the development of innovative and/or complex products and/or services.

The erosion of the value inherent in contractual specifications

Not only is there inherent variability, and therefore uncertainty, in any project for the development of innovative and/or complex products and services, but these projects are also increasingly subject to change from external influences.

The significance of the impact of change on the contractual specifications for products and services cannot be understated. Studies undertaken at the University of Missouri, Kansas City, demonstrate that not only does the inherent value of the specifications for products and services decay over time but that the pattern of the erosion of their value is similar to the pattern of decay exhibited by an unstable radioactive atom. Unstable radioactive atoms decay exponentially. Their rate of decay is described by reference to a 'half-life', which is a measure of the period of time it takes for the substance undergoing decay to decrease by half.[7]

According to the University of Missouri studies, the half-life of the value of a set of contractual specifications for products and/or services has been rapidly decreasing. In 1980 this was around 10–12 years, by 2000 it had fallen to 2–3 years, and it is currently running at about 6 months. In other words, if a contract pre-defines the specifications of products/services, half of those specifications will become obsolete by the end of month 6, half of the remaining half (ie 1/4) will become obsolete by the end of month 12, half of the remaining quarter (ie 1/8) will become obsolete by the end of month 18, and so on. Hence, by the end of month 18, according to the University of Missouri studies, only 1/8 (or 12.5%) of the contractual specifications for the products/services will still possess any inherent value. If the University of Missouri studies are to be relied upon, the implications for commercial contracts are enormous.

This would mean that if a project is running six months late, not only will the return on investment be reduced, but the project is less likely to deliver what the customer actually wants. Even if the project runs according to plan, if the project is scheduled to run for more than a few months the parties have to expect changes.

It is therefore essential that commercial contracts are able to adapt to the current pace of ongoing and continuous erosion of the inherent value in any defined specifications for products/services. However, in the traditional contract models the parties generally have to rely on an upfront description of the products/services in conjunction with the change control mechanism to deal with any changes to that description.

The limitations of the change control mechanism

Unfortunately the change control mechanism is being pushed to breaking point. When the change control mechanism was originally devised, it served a useful purpose in that it identified and segregated changes in the form of change requests. These were worked on separately from the main project, so that the status quo of the main project could be preserved until the impact of the change had been fully analysed and signed off by the parties.

However, what we are seeing today is that projects are subject to so many changes that the scope of the change requests is increasingly wide-reaching, there can be multiple change requests under review at any one time, and there can be change requests to the change requests. The process of analysing the impact of change requests can take so long and be so extensive that it has a destabilising effect on the main team: they are only too aware that their current work may be rendered nugatory following the approval of the change requests. The bigger the proposed change, the longer the hiatus while it is being analysed, and the more damaging the effects can be.

Instead of facilitating change, the change control mechanism actually serves to restrict change. The whole process of documenting changes is time-consuming, consumes valuable resources, can be expensive to implement, and adds no real value to the project. It simply attempts to keep the contract in step with the pace of change.

Furthermore, the change control mechanism actually fosters *'bad behaviour'* between the parties because it polarises their interests. In a fixed price contract it is not uncommon for a supplier to attempt to improve its profit margin by means of inflating the charges for change requests. And the customer is put on the defensive, attempting to justify why a proposed change falls within the existing specifications and does not represent scope creep.

The change control mechanism serves as a distraction to the main project. In any project, no matter how large the organisation and how big the project, only a finite amount of resources will have been allocated to the project. This is for the simple reason that any project has to be justified on a cost-benefit basis. When a change is requested, a member of the team has to be redeployed to analyse its impact. If more change requests are made, the number of team members (or the associated number of man hours) taken off the main project and redeployed to change management activities increases, leaving the main team less well resourced.

Given the wide-reaching impact of many of the change requests, it is not appropriate for the analysis of the change to be segregated and analysed by a small sub-set of the team. The impact of the change should be considered by the team as a whole and its impact considered across all aspects of the solution.

Too often, the change control mechanism results in add-ons without sufficient consideration of which features can be removed and how the overall build can be rationalised. This is because there is very little incentive on the part of the supplier to carry out this exercise. What was once an elegant solution with integrity may evolve into some kind of *'Frankenstein build'*.

This is best illustrated by the development of the M2 Bradley Infantry Fighting Vehicle (IFV) in the US. Originally developed as an armoured personnel carrier, the Bradley, after being subjected to the changing (and often conflicting) demands of a panel of armchair generals, was transformed into a hybrid of a troop carrier, a scout vehicle and an anti-tank weapon platform. Seventeen years later and at a cost of $14 billion the resulting product was *'a troop transport that can't carry troops, a reconnaissance vehicle that's too conspicuous to do reconnaissance, and a quasi-tank that has less armor than a snowblower, but carries enough ammo to take out half of D.C.'*.[8]

In terms of software development, multiple change requests can result in duplications of code and/or conflicts in the code. This in turn can mean that the software is more prone to failure, more expensive to maintain, and that subsequent design and development of the software will be more expensive.

Empirical process control

In projects for the development of complex and/or innovative products and/or services, where the amount of variability – and therefore uncertainty – is significant, it is not practical to work from defined plans. Instead, Scrum advocates the use of empirical process control, that is, a form of control driven by experience and experimentation:[9]

> *'Laying out a process that repeatedly will produce acceptable quality output is called defined process control. When defined process control cannot be achieved because of the complexity of the intermediate activities, something called empirical process control has to be employed.'*[10]

The basic attribute of empirical process control constitutes a continuous cycle of inspecting the process for correct operation and results, and adapting the process as needed. There are three key elements to controlling an empirical process:

- ***Visibility.*** Those aspects of the process that affect the outcome must be visible to those controlling the process.
- ***Inspection.*** The various aspects of the process must be inspected frequently enough so that unacceptable variances in the process can be detected. The frequency of inspection has to take into consideration the fact that the process is likely to be changed as a result of the inspection. The inspector must possess the skills to assess what they are inspecting.
- ***Adaptation.*** If the inspector determines from the inspection that one or more aspects of the process are outside acceptable limits and that the resulting product will be unacceptable, the inspector must adjust the process or the material being processed. The adjustment must be made as quickly as possible to minimise further deviation.

To achieve empirical process control, Scrum establishes an iterative, incremental framework. It splits:

- ***the work:*** into a list of small, concrete deliverables, sorts the list by priority and estimates the relative effort of each;
- ***the time:*** into short fixed-length iterations with potentially shippable code demonstrated after each iteration; and
- ***the organisation:*** into small, cross-functional self-organising teams.

The Evolutionary Contract Model

A new commercial contract model, known as the Evolutionary Contract Model, has been created in conjunction with some of the leading thinkers on Agile and Lean for use in projects for the development of innovative and/or complex products and/or services. The main influences underpinning this model originate in Agile, Lean Software Development (Lean) and systems thinking. The relevant Agile methodologies are Scrum, Extreme Programming (XP), Evolutionary Project Management (Evo) and DSDM Atern.

Unlike the traditional contract model, the Evolutionary Contract Model is not based on defined process control. In other words, it does not place any reliance upon pre-defined plans and specifications. For this reason, there are no specifications or detailed plans in the contract.

The fact that there are no specifications for the solution in the contract leads to a couple of significant consequences. First, there is no need for a change control mechanism, because neither the charging model nor the supplier's focus of work is linked to any contractual specifications. Secondly, there are no contractual acceptance tests because there is no supply of products/services against contractual specifications.

Instead, the Evolutionary Contract Model uses empirical process control to manage complexity, variability and change. The contract sets out the overall scope of the solution. This is expressed in terms of the vision statement for the solution, the product and project goals, and any relevant constraints (such as schedule constraints or regulatory constraints). The concept of the solution is gradually refined, based on the product and project goals and within the parameters of the relevant constraints. As the solution evolves, many things will change along the way. Empirical process control is employed to ensure that the solution evolves within the parameters of the contracted scope.

An overview of the Evolutionary Contract Model

The following overview focuses on the construct of the Evolutionary Contract Model. This overview is merely intended to demonstrate how the principles of Agile and Lean can be reflected in a contract. It does not describe how the Evolutionary Contract Model regulates the full life cycle of the project, nor does it describe how a project which involves multiple and distributed teams should be structured. The Evolutionary Contract Model caters for these possibilities, but they are beyond the scope of this article.

All of the customer's desired features of the solution are captured in a central repository known as the solution backlog. The solution backlog does not form part of the contract and has no contractual status. However, the solution backlog must be within the scope of the contract. The items on the solution backlog (the solution backlog items or SBIs) are prioritised in terms of importance to the customer, and may take the form of products (including software), deliverables or services. The solution backlog may be, and should be, amended and refined by the customer throughout the life of the project. It is this ability of the customer to make changes to the solution backlog at any point in time that is key to building flexibility into the solution.

The development of the solution is conducted in a series of time-boxed iterations of work. The iterations are of fixed duration. They end on a specific date whether the work has been completed or not, and must never be extended.[11] Subsequent iterations build upon the working solution increment produced in earlier iterations.

At the beginning of each iteration the customer selects those SBIs from the solution backlog which are the next most important for the customer. Once the supplier has agreed upon those SBIs which it believes it can complete during that iteration, those SBIs are effectively *'frozen'*. They cannot subsequently be amended by anyone during the iteration, and the acceptance criteria for those SBIs are agreed to by the customer and the supplier before work on the SBIs starts.

Each SBI is defined, built and tested in a fast, concurrent loop. An SBI is evaluated for acceptance, and when it passes the test, another SBI is selected from the solution backlog. If it fails, it is re-worked – on the spot – until it passes the test. The customer assesses whether each SBI is *'done'*, and therefore completed, by checking whether the solution increment that is delivered to the customer at the end of the iteration meets the criteria defined and agreed by the customer and supplier at the start of the iteration. To the extent that the solution comprises software, this means that the code must be fully tested, working and potentially deployable.[12] Warranties could be given by the supplier in terms of the solution increment meeting the pre-defined criteria.

There are a number of different charging models available. Each of these has advantages and disadvantages. The optimum charging model is, to a certain extent determined, by the customer's level of experience of working in an Agile and Lean way and by the level of trust existing between the customer and supplier. Teams that are relatively new to Agile and Lean often choose to use a time and materials model (which provides more accountability than in the traditional contract models); teams that have been working in an Agile and Lean way for some time may choose to use a charging model based on units of work such as story points or function points. More advanced teams are looking to link charges to a quantifiable measure of value.

A team is established by the supplier to develop the solution. The team is both empowered by the customer and accountable to the customer to deliver the project. On the one hand the team has full discretion on how it conducts each iteration, but on the other hand the team is expected to self-organise, self-manage and self-achieve the objectives of the iteration. This means that the team must be cross-functional and must contain a sufficiently wide skill set for the solution to be fully completed by the team without external input. There must be representation from the customer on the team, but the roles of the supplier and the customer are quite different.

Benefits of the Evolutionary Contract Model

The Evolutionary Contract Model reduces the element of uncertainty and therefore risk in the project both by breaking down the solution into many small solution increments and by breaking down the time into short fixed-length iterations. At the end of each iteration the customer is given visibility of the fully completed solution increment. This gives the customer the opportunity and the power at regular intervals to refocus the work of the supplier, and potentially to refine the ultimate solution, based on what the customer has actually seen. This ability of the customer to plan adaptively throughout the term of the project is incredibly powerful.

The Evolutionary Contract Model facilitates a fast and cost-effective development process. The solution increment delivered at the end of each iteration builds upon and is fully integrated with all earlier solution increments. In other words, the solution starts to take shape from the very first iteration and continues to develop from there. At any point in time the partially developed solution will address the customer's most current needs because at the start of each iteration the customer has the opportunity to refocus the efforts of the supplier.

The customer may in fact be able to achieve its objectives for the solution and derive value from the completion of less than half of the features that it originally thought were necessary to build the solution, applying the Pareto principle. According to the Pareto principle (also known as the 80–20 rule), for many events roughly 80% of the effects come from 20% of the causes.[13] It has been demonstrated, for example, in software that 80% of the benefits of an application are derived from the use of just 20% of the features. This is borne out by the results of the Standish Group study, which reported that 64% of software features are typically never or rarely used.[14]

Statistics on traditional and Agile projects

Projects using Agile have been found to be more successful and more likely to be delivered on time than traditional projects.

According to the Standish CHAOS report for 2009 many traditionally developed and run projects were less than successful: 44% were described as challenged and 24% failed. Similarly a report on the failure rates of the US Department of Defense projects in one sample concluded that 75% of the projects failed or were never used.[15]

Those results can be contrasted with the results of several major studies to determine the effects of using Agile methods to manage the development of software. For example, according to an international survey of 4,770 respondents conducted by VersionOne in 2010, 46% of respondents experienced an improvement in their ability to manage changing priorities, and 39% of respondents experienced improved project visibility.[16] On all the measures investigated by VersionOne, at least 94% of respondents said the performance was no worse and, in most circumstances, was improved over the situation before Agile adoption. Similarly, according to an international survey of 131 respondents conducted by Shine Technologies in 2003, 93% of respondents experienced productivity increases, 88% of respondents experienced quality increases, 83% of respondents experienced improvements in customer satisfaction, and 49% experienced cost reductions.[17]

Endnotes

1 '*Apple sells three million iPads in 80 days*' press release on the Apple web site http://www.apple.com/uk/pr/library/2010/06/22ipad.html.
2 '*Nokia loses smartphone market share*' by Andrew Parker, published in the FT on 27 January 2011.
3 Stephen Elop, Nokia chief executive, as quoted in the Times on 10 February 2011.
4 With thanks to Steve McConnell whose book '*Software Estimation: Demystifying the Black Art*' provides the basis for this section.
5 The original conceptual basis of the 'Cone of Uncertainty' was developed by Barry Boehm in 1981. The model has since been validated, based on data from a set of software projects at the US Air Force, NASA's Software Engineering Lab and other sources.
6 According to research conducted by Luiz Laranjeira in 1990.
7 Al Goerner at the University of Missouri, Kansas City.
8 The development of the Bradley M2 fighting vehicle was satirised in the movie clip '*The Pentagon Wars*', available on YouTube http://www.youtube.com/watch?v=pyakI9GeYRs.
9 Scrum is the most popular Agile methodology, according to the survey '*State of Agile Development*' commissioned in 2009 and sponsored by VersionOne.
10 '*Agile Project Management with Scrum*' by Ken Schwaber. Ken Schwaber and Dr Jeff Sutherland are the co-founders of Scrum.
11 There is a variation to this in the Evolutionary Contract Model based upon the principles of Kanban.
12 Although the software is capable of being deployed, the customer may choose not to deploy the software until it is at a greater level of maturity.
13 The Pareto principle was developed by Vilfredo Pareto, a noted economist and sociologist, in the late 1800s.
14 Standish Group study reported at XP 2002 by Jim Jonson, Chairman; internal software products.
15 The US Department of Defense used to be one of the most frequent users of the traditional waterfall methods of development. Throughout the 1980s and into the 1990s most projects run by the US Department of Defense were mandated to follow a waterfall cycle of development as documented in the published standard DOD STD 2167. That is no longer the case. Under the 2010 National Defense Authorization Act President Obama gave Defense Department officials a deadline of July 2010 to create new acquisition processes that can deliver IT systems in no more than 18 months by incorporating certain Agile principles.
16 '*5th Annual State of Agile Development Survey*' conducted by VersionOne, dated 7 November 2010.
17 '*Agile methodologies: Survey results*' conducted by Shine Technologies, 2003.

Conclusion

Unprecedented levels of change arising from the increasing pace of innovation are stretching the traditional contract models to breaking point. Legal practitioners need to find a better way to accommodate change within their contracts. More and more organisations are adopting Agile and Lean principles for the development of innovative and/or complex products and/or services. A new contract model, the Evolutionary Contract Model, based on Agile and Lean principles has been developed, and legal practitioners should consider this as a possible solution. ●

Susan Atkinson is Legal Director at gallenalliance Solicitors.

Gabrielle Benefield is Director at the Scrum Training Institute.

Susan Atkinson and Gabrielle Benefield are currently writing a book on Evolutionary Contract Models, which is due to be published later this year.