Output Based Agreements:
An Agile Approach to Software Contracts

Outsourcing and Trust

Many commentators have suggested that the agile, incremental delivery approach is incompatible with good corporate governance of ICT projects. Indeed, Alistair Maughan, an experienced corporate lawyer who has advised on large public and private ICT contracts including UK HM Revenue & Custom's controversial 10-year £8.5bn deal with Capgemini, has recently argued < http://ow.ly/5Rr1F > that "Agile… won't work in the real world" of government ICT projects. One basic argument used is that projects fail due to a "lack of trust between customer and supplier" and hence the "Agile credo of, "Let's trust each other some more" is undermined from the start.

What do we mean by 'trust' in this context?

The Outsourced IT Experience

Corporate customers have learned not to trust outsourced suppliers. Their experience has been that software projects deliver late, over budget, and represent poor value. They are not looking at the reasons for this; they simply trust their own experience when it comes to negotiating new contracts. So new suppliers start at a disadvantage.

Dialogue and case history helps a supplier build a relationship with a potential customer, so that the client's buying team can feel more confident they will not live to regret the new partnership. Many agile developers feel that the case they make for keeping in tune with the client's needs and delivering to those needs is powerful enough on its own. But purchasers want to know, first and foremost, how much it will cost and what they will be getting for their money.

The business users value what the software does – aka the business outcome. In an integrated Lean value stream, the business users use the software to enhance the value delivered by their business to their customers. But the business users do not commission the software projects. The procurement and retained IT folk who commission projects want assured value for money – they don't want their butts kicked by their senior management for overrunning their budgets. Their performance is measured by compliance to standard practices. Outcomes are a secondary consideration. For them, it is all about managing the cost.

Introducing Agility

Agile teams want to use effective methods to engage closely with the end-users, to deliver value incrementally. But the bigger the customer, the more divorced the procurement and retained IT is likely to be from the business users, let alone the poor forgotten customer. So the buyers design detailed specifications, based on a myriad of unknowable unknowns, and ask developers to price this fiction competitively. Developers, knowing that the specification bears little resemblance to what is actually wanted,

invest money and resources in feel-good relationships and create pricing mechanisms which pay a premium for changes.

It is this whole dysfunctional cycle of 'big design up front' aka 'batch & queue' ineffectiveness that agile developers are seeking to buck. To achieve this revolution in a big business environment, Agile folk need to be able to negotiate meaningful contracts that give the customer assurance not only they will get the desired outcomes, but that they will get it for a given price, within a given timescale.

In the case of software, the question "how much will it cost me" is something like asking 'how much is wood'. But if Agile developers simply answer the question of price with another (e.g. 'it depends on your requirements'), the trust between the parties has not moved forward. The intelligent customer will look for evidence of value for money. Evidence creates trust. Assurance which relies only on words, track record, and the ability of the developer to amass enough supporting evidence to convince the buyer that they will receive value for money seems to me an immensely time-consuming task for the supplier, and provides no concrete assurance to the customer.

The big developers employ teams of salespeople whose role is simply to build a relationship of trust between the client and the supplier organization. The relationship is personal, so the trust established is subjective rather than objective. The cost of this relationship-building is high, and the outcomes are mediocre. It is a system that places effective, efficient, innovative SMEs at a disadvantage and does no favours to the customer. But "no-one gets sacked for buying IBM."

There is dissatisfaction with the results delivered by current methods, but there is little recognition by customers that dysfunctional procurement and contract processes contribute hugely to the problem. The agile SME has to persuade corporate customers to take a new approach to contracting for software services. The key to unlocking the impasse is to divorce the iterative process of exploration and delivery from the process of negotiating the price and terms & conditions.

Software as a commodity

Whatever software developers say about the work they do, those who have responsibility for signing the cheques and commissioning the work do regard software as commodity. Output Based Agreements treat it as such. We know the cost of the software will be determined by the customer's requirements. But scoping the types of software functionality any given customer is likely to want is a relatively easy exercise for anyone with the right know-how and experience. Usually there are only a small number of software 'types', maybe 2 to 8 kinds, each of which will have a different unit price based on the non-functional requirements associated with that 'type' of software. The supplier with a good understanding of their own process costs can easily determine an acceptable unit-price for each of the various kinds of software functionality the customer will need.

In an Output-based Agreement, the parties agree the software type and the required quantity of software functionality for each new development or enhancement project, expressing the scope in terms of the 'functional size' of the requirements. This is where a modern functional size measure such as COSMIC really comes into its own. A simple contractual arrangement can be agreed and the need for detailed specification and prioritisation is deferred, to be thrashed out piecemeal by the end-users and the developers who are closest to the business need (to the 'gemba', as the Japanese say).

Determination of requirements is best described as a process of exploration. This is true for practically all new developments and most enhancements to existing systems. Ideally, it involves both end-users (i.e. experts in the business domain) and technicians (i.e. those with the know-how of what the technology can offer). The requirements evolve as both parties learn more about what is needed, and what is possible, through a process of feedback. And the most pertinent feedback is derived when initial ideas are put into operation. Then people can see how their ideas work in practice. What is more, stakeholders benefit from incremental satisfaction of their highest priority needs.

But this exploration (and experimentation) takes time. So it is in the interests of both parties to get going as soon as possible. No one benefits (except maybe the lawyers) from a long-winded procurement process that involves time-consuming negotiations regarding the exact details of requirements (which is, as I've said, a task doomed to failure). In any case, there is no necessity to determine the Nth level of detail up front, and significant benefits from not doing so. By deferring commitment until the last responsible moment, customers enable suppliers to keep design options open, thereby maximising the potential value delivered.

The technique used by the Output-Based Agreement is familiar to anyone who prepares meals for a family. Say the cook of the family goes to the supermarket for the weekly or monthly shop. They know it is likely their family will want to eat potatoes at several meals during the coming days. But they don't have to decide up front and in exact detail what they will cook. That depends on circumstances and what the family fancies on the day. For different meals they may serve boiled potatoes, mash, roast potatoes, jacket potatoes, potato salad, chips, wedges, potato dauphinoise, etc. (ref: http://www.lovepotatoes.co.uk/recipes). The choice of what to cook can be deferred to be decided until just before each meal. All that's written on the shopping list is, "potatoes" and maybe "oven chips".

So the cook buys a quantity of potatoes, say 5 kilos. For an agreed price. Say 78p per kilo. A total cost of £3.90 GBP. Of course, every potato is different. Some are small, some are larger, some are a funny shape. It doesn't matter. It is perfectly feasible for the shopkeeper and the shopper to negotiate a price per kilo and clinch the deal.

That's essentially how an Output-Based Agreement works. The customer and supplier agree delivery of a quantity of software functionality, and a price that is satisfactory to both. The total price is calculated from an initial coarse estimate of the quantity required, say 1000 COSMIC Function Points (CFP), and an agreed unit price, say £500 GBP/CFP. So the contract price is agreed at £500,000 GBP.

The customer knows what their budget is; the supplier likewise must ensure they negotiate a fair price which allows for a suitable profit margin. This relies on having a good idea of the process performance they achieve, and hence their unit cost for producing software (of the kind required, using reasonably familiar technology, etc.). Given such information, and agreement from the customer to commit end-user effort to the development, they will be able to commit to agreed completion dates. They'll be able to determine how many teams, the team size, and number of iterations or sprints.

Experience suggests that, because there always will be some uncertainty in the initial estimate of the quantity required (i.e. the functional size of the functional user requirements), it is wise for the parties to agree a tolerance (say +/– 10%) for the total delivery.

The OBA can include clauses that cover the situation for when the scope of the functional user requirements turns out to be

larger than both the initial estimate and the tolerance combined. One way is to agree a 2nd tolerance band of another 20% say, to permit a total requirements size of up to 1000 x 130% = 1300 CFP. I suggest a premium price is agreed for these 'extra' requirements, say £600 GBP/CFP, in this example. This is to dissuade the customer staff from gold-plating their requirements unnecessarily. The funding for such 'extra' requirements should come from a 'risk reserve' budget, managed at a senior level, so that the Product Owner and the Development Team have to justify the additional expenditure. By directly linking requirements to costs, the customer is constrained to manage requirements intelligently and collaborate on managing scope and costs. The supplier cannot ramp up unnecessary costs on make-work.

A further clause can be included in the OBA to cater for the situation when, during the early exploration of the desired outcome, the parties determine that the initial coarse estimate is seriously flawed. In which case, the best thing to do may be to stop and start again.

Output-based Agreements satisfy both the customer's need for an assured outcome, and the suppliers desire to use effective lean-agile methods. They introduce a need for measurement discipline, which in turn fosters more openness and honesty in outsourcing partnerships. They also allow like-for-like price comparison for commodity software.

It is taken as read that all car insurers will offer you the same basic commodity - they'll all offer you 3rd party, fire and theft with a no claims discount. But like for like, who gives you the best deal? Tescos and Sainsburys will both stock a range of baked beans and potatoes. Both will say, if you want potatoes, we provide best quality potatoes at highly competitive prices. If you want a ready meal, our gourmet platters are the best there is. They compete by providing stock items at competitive prices, and add value by providing other in-store services. The same approach can be taken by software developers, producing basic functionality at a visibly competitive price and leveraging the iterative agile approach to focus on delivering the right outcome for the customer. *'This is not just software…it's flexible, outcome-focused software'*!

However, as such 'open book' accounting of software productivity will almost inevitably favour the smaller, more efficient software houses, there is little interest in objective cost measurement from the established players. It is down to the innovative players to push for better and more effective ways of contracting software services.

Output-Based Contracts have been used by some since the 1990s. The comparative measures of functionality which are used to size the output have been significantly refined since their origins in the 1980s, and modern methods map easily to Agile approaches to scoping and managing delivery schedules, budgets and resourcing, adding a necessary degree of business focus to activities such as estimating and scope management. Contracts such as these recognise the uncertainty inherent in development and innovation, while providing the decision-makers on the customer side with certainty on their critical requirements. What does it cost, what am I getting for my money, will I live to regret this deal? The Project Sponsor gets assured delivery of an agreed scope, on time, within budget. They understand the price and can demonstrate value-for-money. The Development Team and the Users work together effectively to explore the demand & deliver results incrementally. Everyone's happy.

Simple really. But isn't that the essence of the lean, agile approach?

Grant (PG) Rule
Executive Coach, Orchestrated Knowledge Ltd.

St. Clare's
Mill Hill
Edenbridge
Kent
TN8 5DQ

T: +44 1732 863 760
M: +44 7770 503 241
E: g.rule@orchestratredknowledge.com