Contracting for Agile Software Projects Part 1, by Peter Stevens

As a customer or supplier of software services at the beginning of a Software Development Project, you know that there is too much at stake to work with just a verbal agreement. Although the Agile Manifesto values customer collaboration above contracts, contacts are necessary when working with external suppliers.

A contract is really just a set of written playing rules. The right rules increase the chance of success for both parties. The wrong rules make cooperation difficult and hinder progress.

This is the first of two articles. In this article, we'll look at the purpose and contents of a contract and some criteria for evaluating agile contracts. Next week, we'll look at contracting alternatives and identify their strengths and weaknesses for agile development projects.

What is the purpose of a contract?

Contracts set the basic playing rules for the project. In theory, they are freely entered by both parties to create optimal conditions for successfully completing the project. In practice, contracts are often seen as competitive games, in which the objective is to place the other party at a disadvantage, especially if things go badly. Very large companies and governments often have standard conditions which have to be accepted en bloc as a pre-requisite to doing business with them. These conditions are seldom fair, so a reasonable project outcome depends heavily on a good relationship with the actual customer and avoiding recourse to the contract or the law. (The Agile Manifesto is right in this point: customer relationships are more important than written contracts!)

Even negotiated contracts do not always strive for a win-win situation, so you may need the help of experts. However from the customer perspective, contracts produce no added value. They are a waste product, so the effort spent negotiating & producing them should be minimized.

A contract apportions risk and reflects trust between the parties. What happens if something goes wrong? Who pays how much if the project is more difficult than expected? Who benefits if the project is finished earlier than planned?

The wrong playing rules can be detrimental to the success of the project. Bad rules can lead to unrealistic prices, time frames or functional expectations. Win-lose games are detrimental to project success. Quality most often suffers. Do you want the 'A-Team' or the 'B-Team' working on your project? Think carefully about how much pressure you put on the price.

What information should a contract include?

The more trust that exists between customer and supplier, the less you will need to write down. In my experience, there are a couple of points which belong in every contract:

- Objectives of the project and of the cooperation between the companies. This follows pretty directly from the elevator pitch and product mission.
- An outline of the project structure - Scrum process, key roles and any differences from Scrum which apply.
- Key Personnel - who is responsible at the operational and escalation levels and what is required of these people?
- Payment and billing, including any bonus and penalty clauses
- Early and normal termination.
- "Legal Details." Depending on local law and legal customs, you may need to exclude liability, specify venue, ensure severability (that portions of the contract are remain in effect, even if parts of the contract are found invalid) or include other text to prevent various legal bad things from happening. Sample or reference contracts from your jurisdiction can be helpful (and cheaper than a lawyer!).

Do you need to include the scope in the contract? Often it is present (at least in the government contracts I've had the privilege signing, it was), but fixing the contract in scope also renders scope inflexible. You may want to specify how you will manage the scope (e.g. Product Backlog, Sprint contract), but operational details should be left to the project team.

Contracting for Agile Software Projects, Part 2, by Peter Stevens

As a customer or supplier of software services at the beginning of a Software Development Project, you know that there is too much at stake to work with just a verbal agreement. A contract is really just a set of written playing rules. The right rules increase the chance of success for both parties. The wrong rules make cooperation difficult and hinder progress.

Last week, we looked at the purpose and contents of a contract and some criteria for evaluating agile contracts. Today, we'll look at contracting alternatives and identify their strengths and weaknesses for agile development projects.

Contract Forms

Commercial contracts can take many forms. What are the contract alternatives that are suitable for agile development projects? For each of the contract forms below, I look at:

- How is the contract structured?
- How does it apportion Risk and Reward between customer and supplier?
- How does it handle changes in requirements?
- What model of customer relationship does it foster: competitive (my win is your loss), cooperative (win-win), indifferent (I don't care-you lose) or dependent (heads-I-win-tails-you lose)?

Sprint Contract

This is not really a contract, but I have found the metaphor of a "Sprint Contract" to be helpful in understanding (and sometimes enforcing!) the relationship between product owner and implementation team. It is merely the agreement between the Product Owner and the Team for one sprint.

The implementation team agrees to do its best to deliver an agreed on set of features (scope) to a defined quality standard by the end of the sprint. (Ideally they deliver what they promised, or even a bit more.) The Product Owner agrees not to change the project before the end of the Sprint. So a Scrum project simply is a series of mini projects with fixed parameters: Time (Sprint Length), Scope (Sprint Backlog), Quality (Definition of Done) and Cost (Team Size*Sprint Length).

The Sprint contract can be referenced in the commercial contract. I have found that after a couple of releases, the commercial contract withers down to a one page time & materials agreement, maybe with a cost ceiling for the quarter or next major release.

Time and Materials

How: Work for a month, then send the customer an invoice. Suppliers like it, because the customer carries the risk of changing his mind.

Risks: carried 100% by the client. Supplier has little incentive to keep costs down. Effort to ensure that only legitimate effort and expenses are invoiced can be substantial.

Relationship: Indifferent. The supplier is happy when more work comes because more work means more money.

Tip: recommended for projects where the customer can better manage the risk than the supplier. This is often combined with a cost ceiling (although this can degenerate into a “heads I win, tails you lose contract” if too much pressure is applied on the price).

Fixed Price / Fixed Scope

How: Agree on the deliverables, deliver it. Send a bill. Customers like fixed price projects because it gives them security (or at least they think so).

Risk: Obvious risk is on the side of the supplier. If the estimates are wrong, the project will lose money. Less obvious risks are the change request game, through which the supplier negotiates additional revenue through scope changes. If the supplier had badly underestimated the effort or quoted an unrealistically low price, the losses can threaten the existence of the supplier, which also presents a problem to the customer.

Relationship: Competitive to indifferent. Customer generally wants to have more and the supplier wants to do less. The supplier wants the customer to be happy, so usually the supplier yields. The word ‘et cetera’ is very dangerous word in a specification to a fixed price.

Tip: Specify the functional requirement with user stories. I’ll discuss strategies for hitting the target on a fixed price project in a future article.

Time and Materials with a Cost Ceiling

How: Same as time and materials, but the cost ceiling limits the risk of the customer. Generally the scope is fixed.

Risks: This represents the ‘best of both worlds’ from the customer’s point of view. If it requires less effort than expected, it costs less. But the once the cost ceiling has been achieved, it behaves like a fixed price project.

Relationship: Potentially cooperative, but more likely competitive or even indifferent, depending on how aggressive the cost ceiling is. From the supplier’s point of view, this is a ‘heads you win, tails I lose’ project. The goal is to hit the cost ceiling exactly. There is no incentive for the supplier to deliver below the maximum budgeted cost.

Tip: If the scope is allowed to vary, this can be an excellent basis for cooperative development.

Phased Development

How: Fund quarterly releases and approve additional funds after each successful release.

Risk: Customer’s risk is limited to one quarter’s worth of development costs.

Scope Changes: new business goal for each release, compatible with a product backlog prioritized by the Product Owner.

Relationship: Cooperative. Both the customer and the supplier have an incentive that each release be successful, so that additional funding will be approved.

Tips: Venture capitalists often work on this basis. Scope is flexible, at least at the release level. Each quarter is a new planning and funding round. Often a cost ceiling per phase is defined. This is usually not seriously limiting

I have worked quite happily under this model. We simply specified the Release goal, hourly rate and cost ceiling in the commercial contract. The customer provided the product owner. Everything else was determined in the sprint contracts.

Bonus / Penalty Clauses

How: Supplier receives a bonus if the project completes early and pays a penalty if it arrives late. The amount of bonus or penalty is a function of the delay

Risk: Does the customer have an incentive for early completion? The ROI arguments must be compelling and transparent. Otherwise the customer gets a cheaper solution the longer it takes.

Scope Changes: difficult to accept because changes potentially impact the delivery date, which is surely not allowed.

Relationship: could be cooperative, but might degenerate into indifferent if the customer does not truly need the software by the date agreed.

Tip: Often applied for construction projects, e.g. roads, tunnels and runways, for which it works well. Scope changes are not issue and genuine economic costs drive the customer to achieve the deadline as well.

Fixed Profit

How: any project budget consists of effective costs and profit. The parties agree on the profit margin in advance e.g. 20%. The customer can back out after “any sprint” by paying a cancellation fee of profit part of the remaining effort.

Scope is variable - it is possible to finish early. This only works with agile development, because only agile methods deliver software incrementally.

Risk is shared. If project finishes early, the customer pays less, but the supplier still has his profit. If the project exceeds budget, the customer pays more, but the supplier does not earn additional profit. After the target delivery date, the supplier may not invoice any more profit, just cover his costs.

Relationship: Cooperative - both have a clear incentive to finish early. The customer saves money and the supplier has a higher profit margin.

Tip: This is often combined with a variable scope contract to form the “Money for nothing, changes for free” contract. This is

“Money for Nothing, Changes for Free”

How: This is a combination of several approaches. Work is basically on a Time and Materials basis with a cost target. However the intention is that the project will not use up the entire project budget. After a certain amount of functionality has been delivered, the customer should realize that enough business value has been realized that further development is not necessary and can therefore cancel the project. A cancellation fee equal to the remaining profit is due.

Risk: Shared. Both parties have an interest in completing the project early. Customer has lower costs, supplier has a higher margin.

Scope: can be changed. Planned but unimplemented features can be replaced with other stories of the same size. Additional features cost extra.

Alternatives: If the budget is exceeded, the rules of the fixed profit or cost ceiling contracts can be applied. The fixed profit approach is more consistent with the goal of a fostering a cooperative relationship.

Joint Ventures

How: Two partners invest in a product of joint interest.

Risks: two of everything. Decision chains can get long. Rivalries can develop between the teams. Different models for extracting value from product can lead to different priorities for further development.

Tips: treat the project as a separate company: One team, co-located, with development and product marketing /product owner role. Think realistically about friendly and not so friendly separation scenarios.

Recommendations

I have worked quite happily for years with a phased development contract. The original contract was a fixed scope contract with a cost ceiling, but as we worked together and built up the level of trust, the surrounding text just withered away. Trust, a bit of boilerplate, the sprint contract and a quarterly sign-off from top management worked quite nicely.

I think the “Money for nothing, changes for free” contract could be very attractive. It’s a cooperative model, which offers incentives to both parties to keep the costs down and rewards higher productivity achieved with Scrum teams. It might be compatible with a cost ceiling although I would be wary of losing the cooperative relationship.

The contract form lays the important groundwork for a successful project. And the Agile Manifesto got right: working with the customer is more important than the contract. So whatever you do, keep the customer relationship positive!