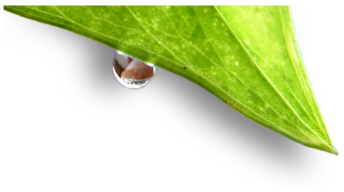


Ten Contracts for your next Agile Project, by Peter Stevens

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, contracts 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. Which contract forms are best for agile software development projects?

First, let's look at the purpose and contents of a contract and some criteria for evaluating contracts in an agile context. Then we'll look at contracting alternatives for Scrum software projects and examine their strengths and weaknesses.

What is the purpose of a contract?

Contracts set the basic playing rules for the project. In theory, both parties freely enter the contract to create optimal conditions for successfully completing the project. In practice, contracting is often seen as a competitive game, 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 that must 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 supplier.

How to evaluate contract forms

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

- How is the contract structured? What are the basic rules for delivering scope and invoicing revenue?
- 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)?

If I did not draft the contract, I would be on the lookout for traps: contact negotiations are often competitive games. The question is what do you do if you find one? Try to get it taken out, or accept it and move on? Let's just call this a 'business decision'.

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 that belong in every contract:

1. Objectives of the project and of the cooperation between the companies. This follows pretty directly from the elevator pitch and product mission.
2. Project structure. It may be sufficient to say you will do Scrum, or you may want to include a description of Scrum.
3. Key Personnel - who is responsible at the operational and escalation levels and what is required of these people? For example, who will fill the roles of ScrumMaster and Product Owner?
4. Payment and billing, including any bonus and penalty clauses
5. Early and normal termination.
6. "Legal Details." Depending on local law and legal customs, you may need to limit civil 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 are cheaper than a lawyer!).
7. Other stuff – as appropriate for your context.

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 included by law), but fixing the contract in scope also renders scope inflexible. If possible, it is better to specify how you will manage the scope (e.g. Product Backlog, Sprint contract), but operational details should be left to the project team.

Points 2, 3, 4 and 5 determine the playing rules for your project. If you get these right, you will have the foundation for a good project. But what are the best rules? There are many different kinds of contract, from time and materials to fixed price, fixed scope.

Playing Rules for Software Development

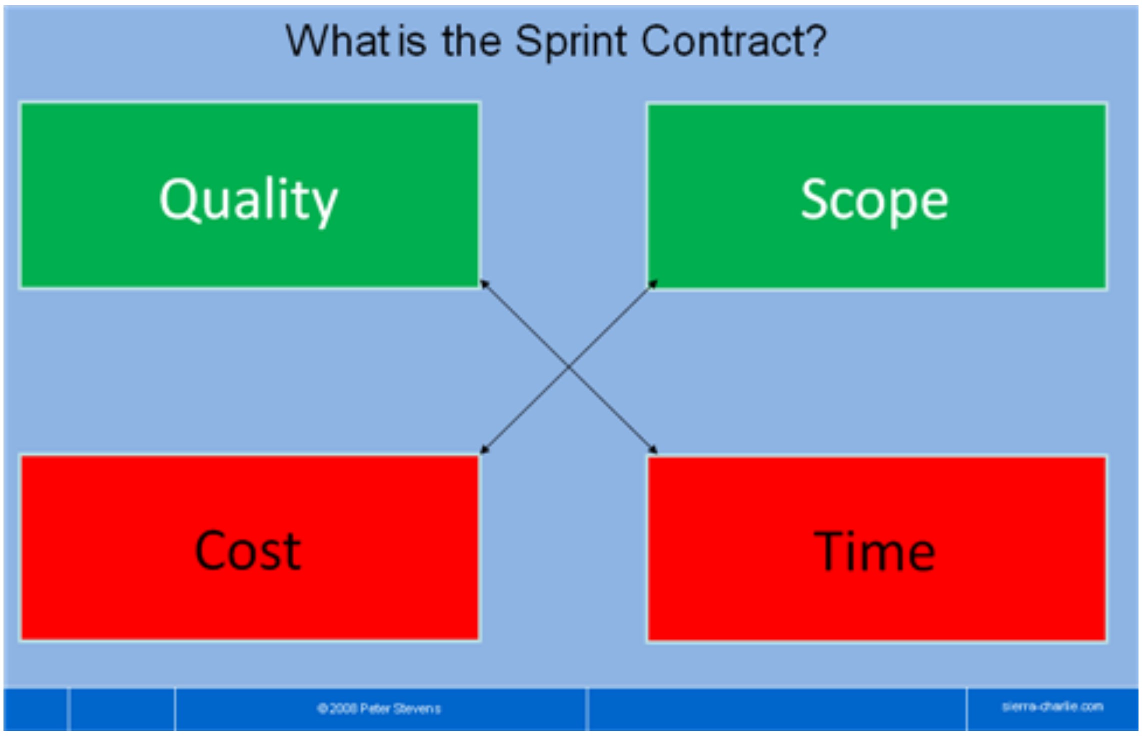
What are the available playing rules and what is the best approach for a agile project? This section examines 10 different contract forms and evaluates according to the following criteria:

- How is the contract structured?
- How does it handle changes in scope (requirements)?
- How does it apportion Risk and Reward between customer and supplier?
- 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)?

Let's look at a number of possible contracts, one at a time, and see how they work with agile and Scrum development projects:

1. the "Sprint Contract"
2. Fixed Price / Fixed Scope
3. Time and Materials
4. Time and Materials with Fixed Scope and a Cost Ceiling
5. Time and Materials with Variable Scope and Cost Ceiling
6. Phased Development
7. Bonus / Penalty Clauses
8. Fixed Profit
9. "Money for Nothing, Changes for Free"
10. Joint Ventures

Sprint Contract



Working with Scrum, 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.

Structure: This is not really a commercial contract, but simply the agreement between the Product Owner and the Team for one sprint.

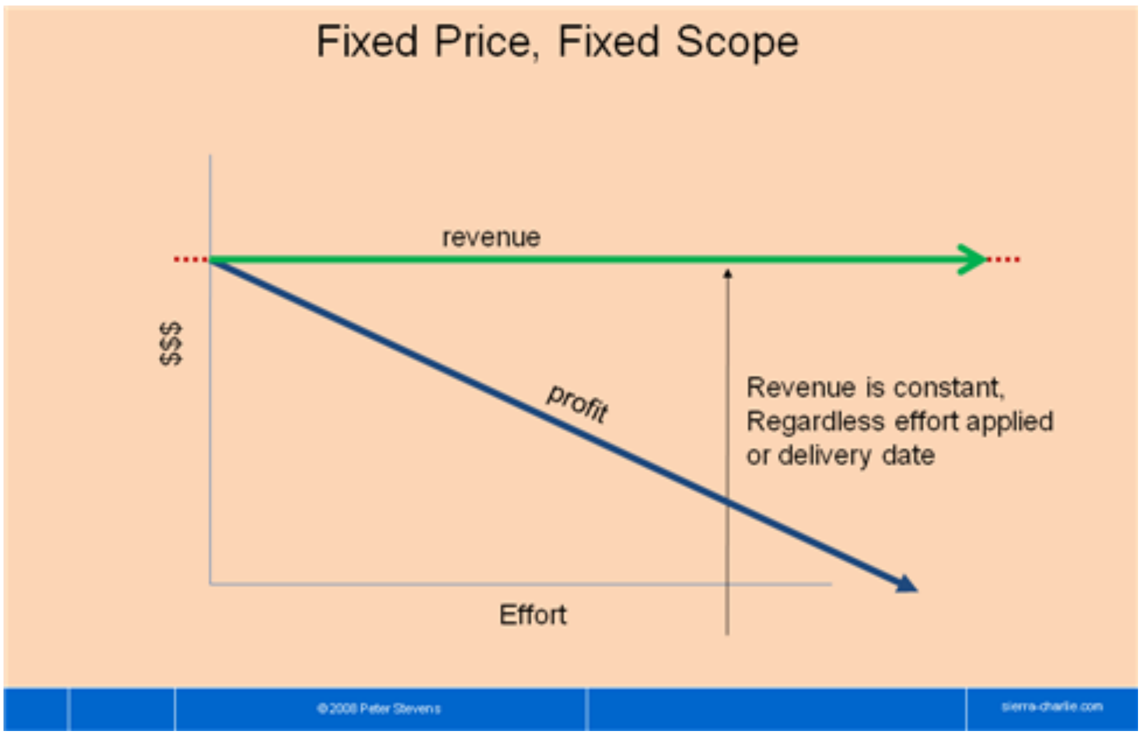
Scope: The implementation team agrees to do its best to deliver an agreed on set of features (scope) to a defined quality standard by the end of the sprint. (Ideally they deliver what they promised, or even a bit more.) The Product Owner agrees to help the team as best s/he can and not to change the scope before the end of the Sprint.

Risk: a Scrum project can be considered 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). Only the scope is allowed vary; Team and Product Owner measure progress every sprint based on completed scope. Short sprints limit risk and enable rapid response to real issues.

Tip: Confirming Sprint Contract in via E-Mail or posting it on the project Wiki at the beginning of every Sprint is usually good idea. It builds trust, regardless of the underlying contractual form.

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

Fixed Price / Fixed Scope



Structure: 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).

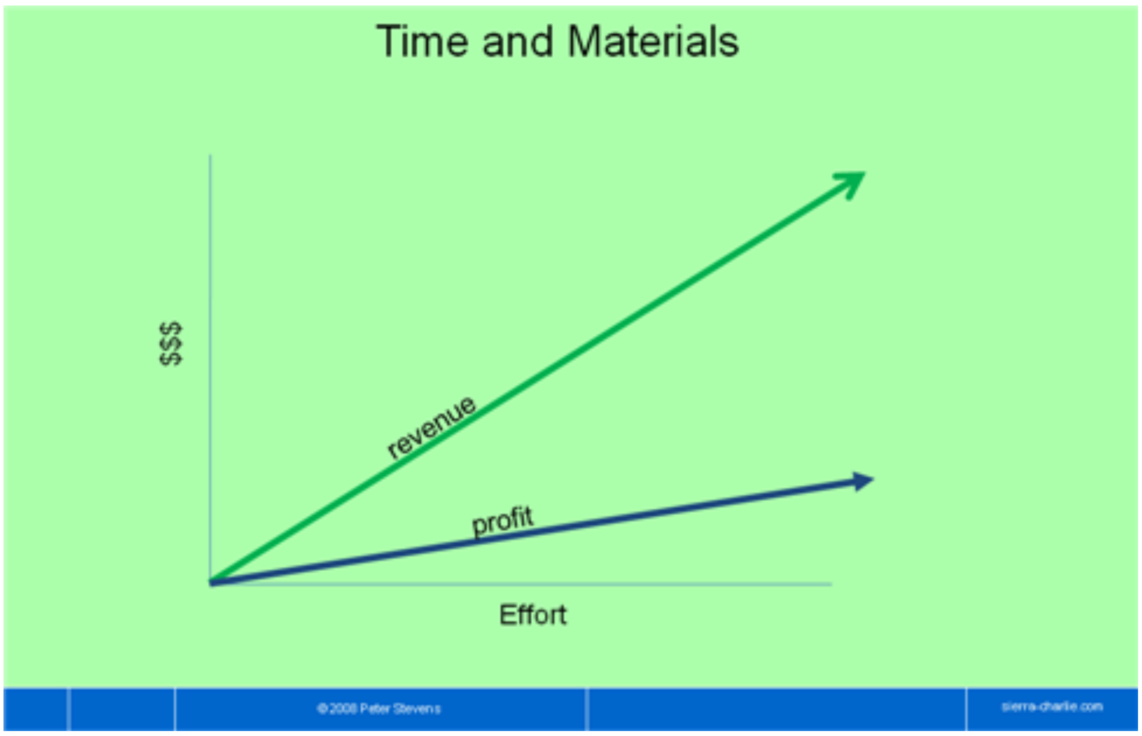
Scope changes: The name says it all, doesn't it? The change request game (correction: change request process) is intended to limit scope changes. This process is costly, and the changes are usually not preventable. Since the customer almost by definition wants more scope, ending the project can be difficult. The supplier wants the customer to be happy, so the supplier usually yields. The words 'et cetera' are very dangerous in the specification of a fixed price requirement.

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 risk, or quoted an unrealistically low price, the losses can even 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.

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



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

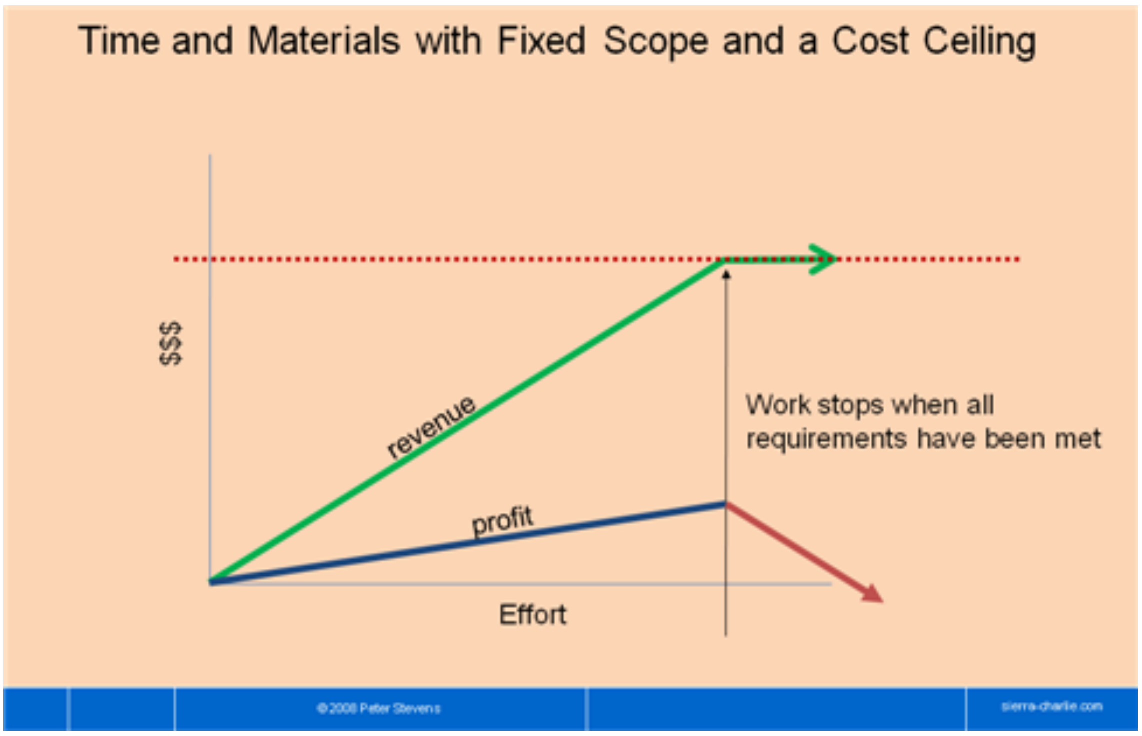
Scope: Not a priori fixed. Sooner or later, the customer doesn't want to pay any more, so the project comes to an end.

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. How well it works can depend on how the scope is handled.

Time and Materials with Fixed Scope and a Cost Ceiling



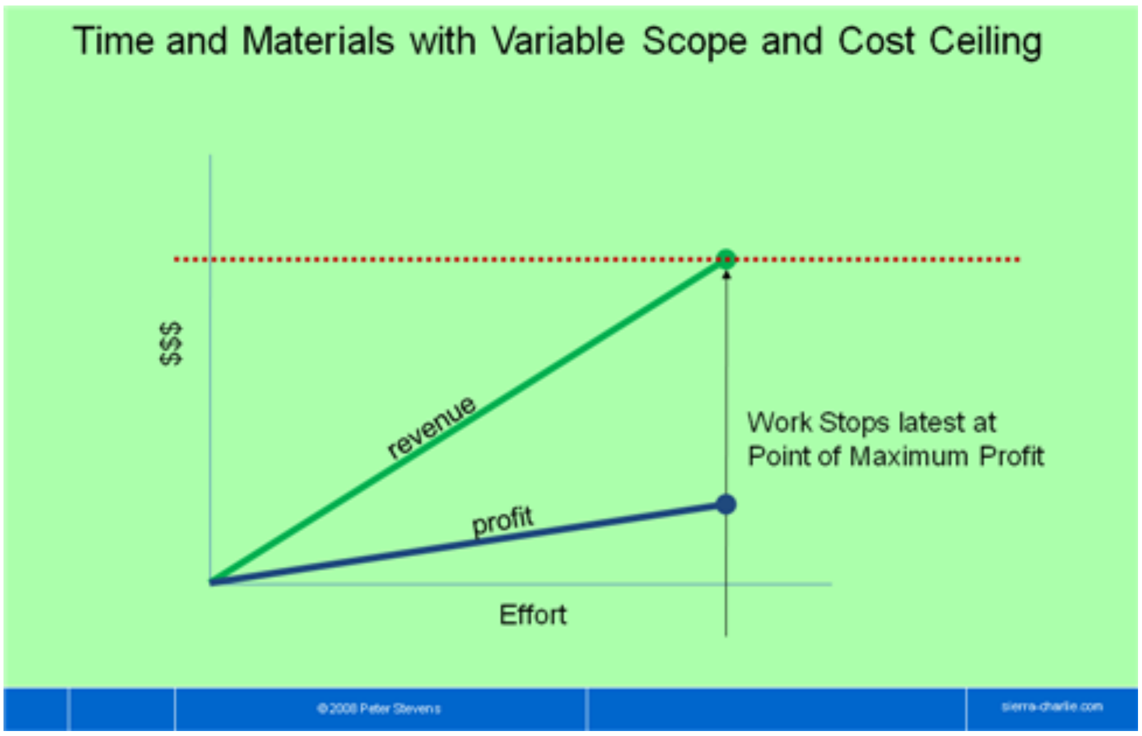
Structure: Same as fixed price, fixed scope, except if the vendor gets finished early, the project costs less, because only actual effort is invoiced.

Scope: Same as fixed price, fixed scope.

Risks: This appears to represent 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: Dependent. From the supplier's point of view, the goal is to hit the cost ceiling exactly. There is no incentive for the supplier to deliver below the maximum budgeted cost. The customer has probably treated the project internally as a fixed price project, and so has no incentive little renounce scope to save money.

Time and Materials with Variable Scope and Cost Ceiling



Structure: Same as time and materials except a cost ceiling limits the financial risk of the customer.

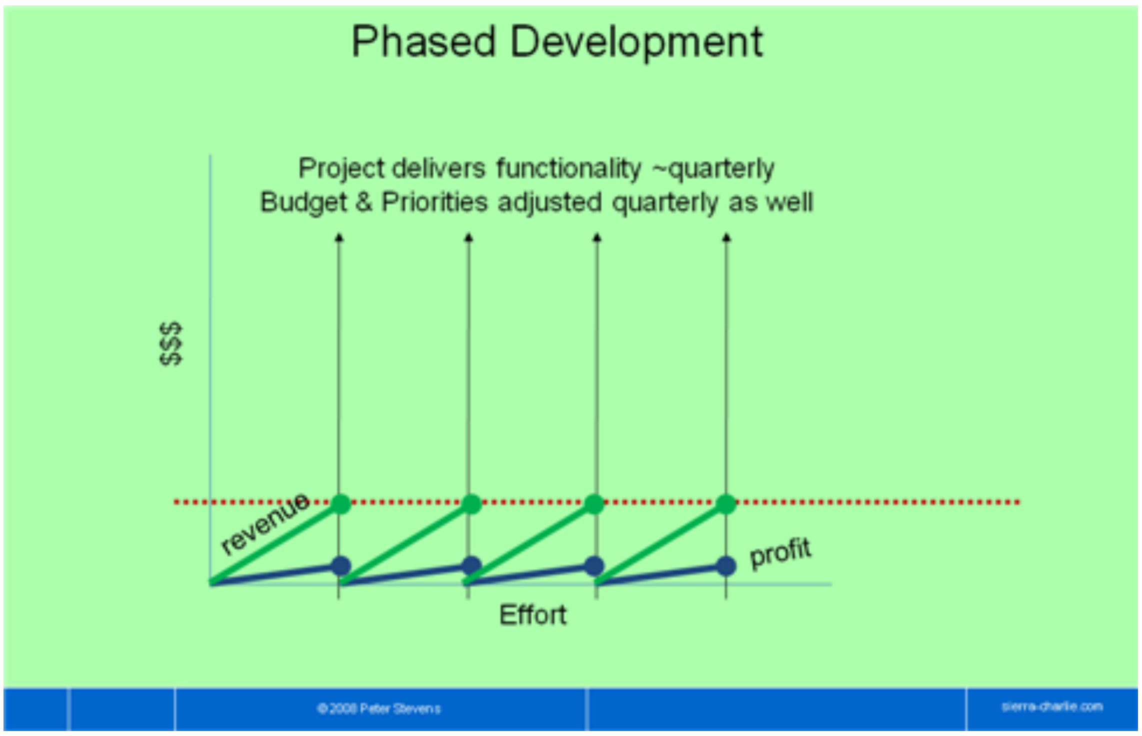
Scope: Same as a fixed price, fixed scope project.

Risks: the budget will expire without achieving the necessary business value for the customer. Customer won’t get everything he asks for.

Relationship: Cooperative. The combination of limited budget and variable scope focuses both customer and vendor on achieving the desired business value within the available budge.

Tip: From experience, I would say this contract form mixes well with Scrum. The customer has control over each individual sprint. A constructive relationship means that even if problems develop on the way, the emotions are right to come to a mutually agreeable solution.

Phased Development


Phased Development
Project delivers functionality ~quarterly
Budget & Priorities adjusted quarterly as well
$$$
revenue
profit
Effort
©2008 Peter Stevens
sierra-charlie.com

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

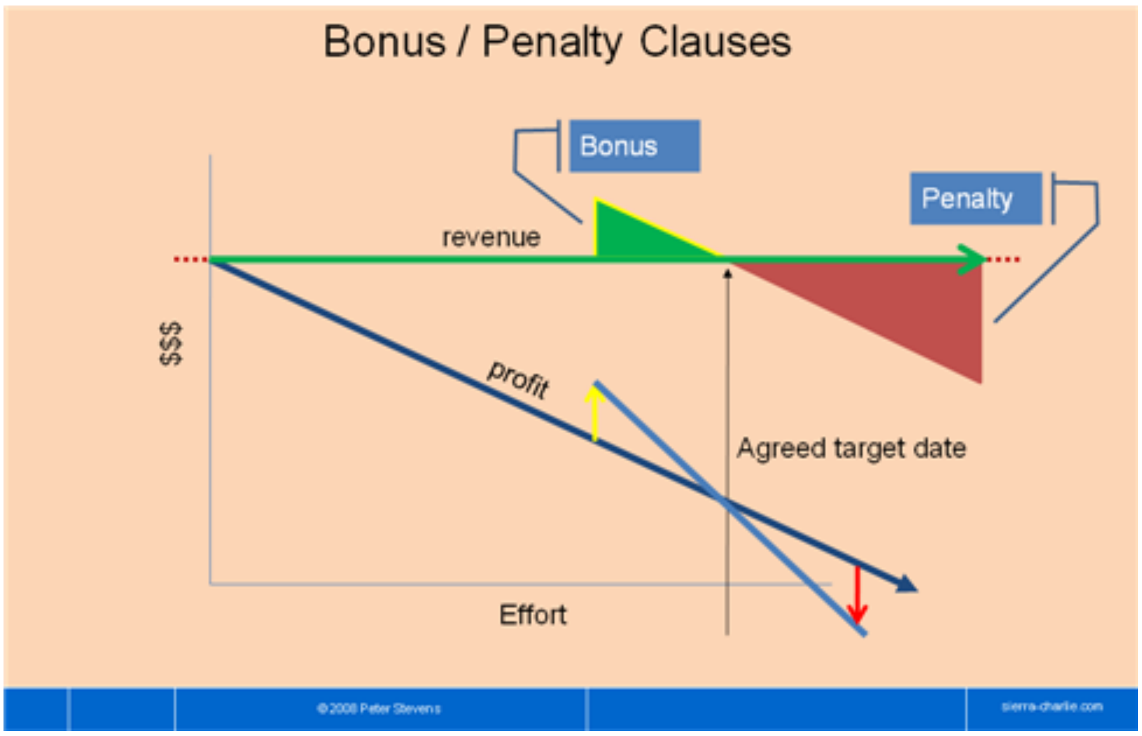
Scope Changes: Not explicitly defined by the model. Releases are in effect time boxed. The knowledge that there will be another release next quarter makes it easier to accept postponing a feature to achieve the time box.

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

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. This mixes well with Time and Materials with Variable Scope and a Cost Ceiling. 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



Structure: 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

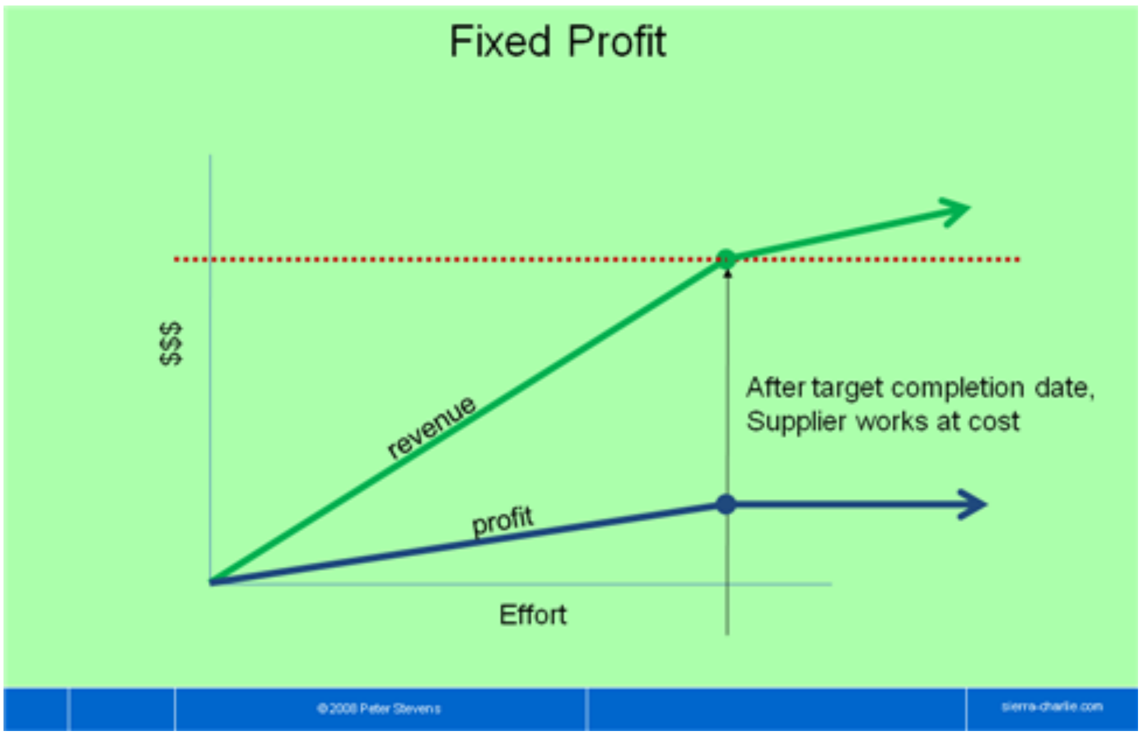
Scope Changes: difficult to accept because changes potentially impact the delivery date, which is surely not allowed.

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.

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



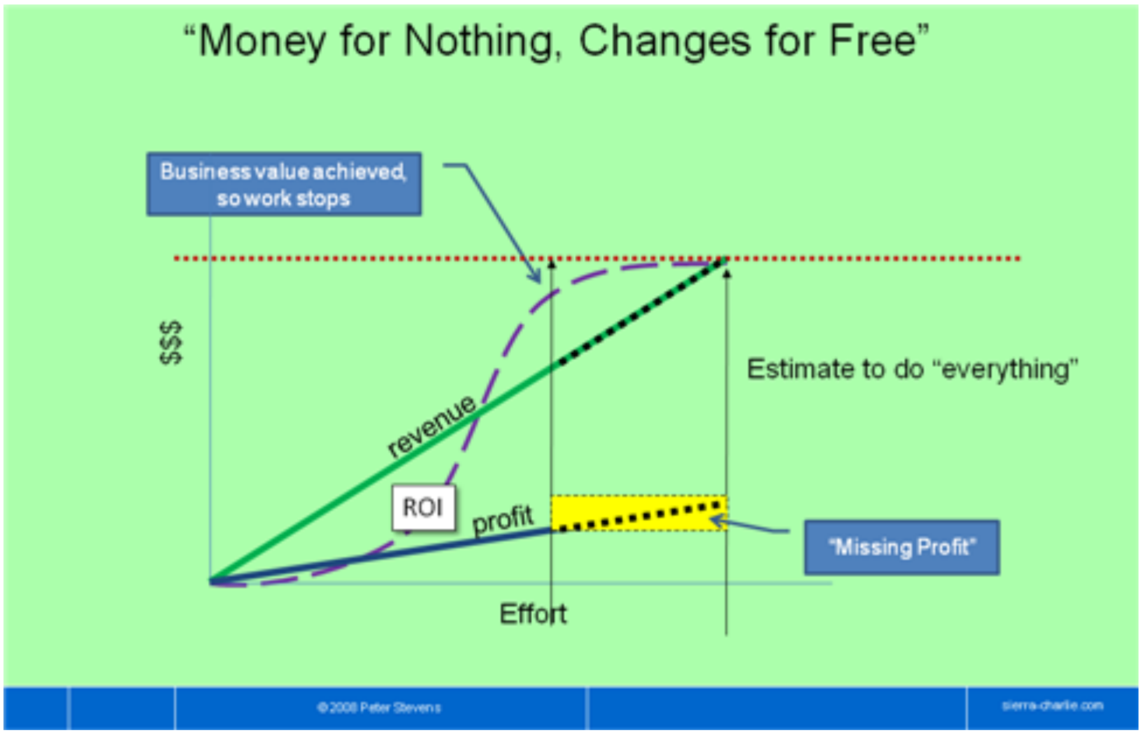
Structure: any project budget consists of effective costs and profit. The parties agree on the profit in advance, e.g. $100,000. Regardless of when the project is completed, the contractor receives the incurred costs plus the agreed profit.

Scope: is fixed

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.

"Money for Nothing, Changes for Free"


"Money for Nothing, Changes for Free"
Business value achieved, so work stops
$$$
Estimate to do "everything"
revenue
ROI
profit
"Missing Profit"
Effort
© 2008 Peter Stevens
sierra-charlie.com

Structure: This works with agile software projects because there is little or no work in progress. After each sprint, functionality is either complete or not started. Work is basically on a Time and Materials basis with a cost target, often with the intention that the project should 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.

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

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

Tip: 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

Photo courtesy of hydropeek@flickr

Structure: Two partners invest in a product of joint interest. Money is unlikely to flow directly between the partners in the development phase, but each party must have an ROI, which may come from revenue sharing or just benefits from using the software.

Scope: Defined to suit the needs of the partnership.

Risks: Two of everything. Decision chains can get long. Rivalries can develop between the teams. Different models for extracting value from the product can lead to different priorities differing willingness to invest.

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.

“Money for nothing, changes for free” contract turns the advantages of the Scrum and agile development processes into a competitive advantage. By prioritizing and delivering business value incrementally, you can dramatically reduce the chances of an outright failure. This advantage is passed on to the customer. Furthermore, it’s a cooperative model, so it offers incentives to both parties to keep the costs down.

The early cancellation clause rewards the higher productivity achieved with Scrum teams. On the down side, this clause feels a bit like a 'golden parachute' which may not be politically acceptable. ‘Money for nothing’ might be work with a cost ceiling although I would be wary of losing the cooperative relationship.

Conclusion

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!

Further Reading

Fixed price projects and agile development are considered not to mix. I know from experience that this is not strictly true. I am also convinced that other forms are better. http://bit.ly/eGXXR3

A fixed profit contract is also known as Cost Plus Fixed Fee - there are other several variations on Cost Plus contracts. http://bit.ly/gXrxIf

Excerpt on Contracts from Lean Software Development, by Mary Poppendieck and Tom Poppendieck - a summary of many current contract forms, a discussion of trust, and a reminder that there is more to success than satisfying the 'iron triangle.' http://bit.ly/fkn8Ei

Jeff Sutherland's Money for Nothing: http://bit.ly/f2UWeb