

# Collaborative Agile Contracts

Lars Thorup  
BestBrains ApS  
lars.thorup@bestbrains.dk

Bent Jensen  
BestBrains ApS  
bent.jensen@bestbrains.dk

## Abstract

*We report on our experiences from two commercial projects conducted under a new form of contract that supports agile development and encourages efficient collaboration between customer and supplier. We show actual extracts from our contracts and describe how to adjust a contract to specific project conditions.*

## 1. Introduction

BestBrains, a consulting company from Copenhagen, Denmark, used to provide development assistance as a time-and-material subcontractor to a number of projects. These projects were normally run under a fixed-scope-fixed-price regime, and we had seen the consequences in the form of low trust, blaming mentality and other kinds of suboptimal behavior that ultimately lead to poor results and delayed delivery.

One year ago we decided to enter into a new business area where we take responsibility for delivering complete solutions to our customers. In order to be able to work with our customers in a collaborative agile setup and thus avoid the dysfunctions we had experienced in other projects, we decided to evolve a type of contract that will:

- Allow the supplier to develop the solution iteratively and settle on details incrementally.
- Keep a high level of development quality throughout the life of the project.
- Give customer and supplier an incentive to work collaboratively and strive for solutions that are advantageous to the project as a whole.
- Give customer and supplier an incentive to finish the project early with the right amount of functionality.

We are developing and improving this type of contract through a number of real-life projects. At the time of writing we have experience from two such experiments.

The first experiment was conducted when we in August 2008 started working with a small event bureau on an interactivity platform. The second experiment was started

when we took over the second phase of the development of a power plant registration system for a large Danish energy corporation. The energy corporation were very frustrated with the quality of work of their former supplier, a large IT-consulting company that used a strict waterfall-approach under a fixed-price-fixed-scope contract. Their frustration meant that they were willing to try something new to improve the result of the second phase, and it was not hard to persuade them to try an agile approach, and start work on finding a contractual basis for this.

## 2. Why do we need agile contracts?

An important cornerstone of agile development is to settle on details during the course of project and defining scope on a per-iteration basis. The purpose is twofold. Firstly it minimizes risk, since decisions are made when more knowledge is available. Secondly, but not less important, it gives room for adjustments based on feedback during development. These benefits will yield a product that is more likely to fulfill its purpose and suit the needs of users.

When a software project is conducted in a commercial setting with one company as the customer and the other as supplier, there is usually a contract between the two governing rights and obligations. An important purpose of all contracts is to define how risks are balanced in the project.

The standard contract with fixed-scope-fixed-price is putting all risk on the shoulders of the supplier [1]. The experienced (or lucky) supplier knows this and makes sure to have room for risk in his bid, and also cleverly plays the change-control game, where anything not mentioned in the specification is to be paid for separately. Other less experienced (or less lucky) suppliers are caught by too low estimates, unclear specifications and an inability to play the change-control game. The latter type of supplier sees a declining profit, and often then (whether deliberately or not) uses quality as a parameter to adjust the efforts.

Another alternative is time-and-material contracts where the supplier is paid by the hour. That allows for a project where the software is developed iteratively and

where scope is not mentioned in the contract. With this kind of contract the customer assumes most of the risk [1]. The supplier has not committed to anything but to deliver a certain amount of hours, and maybe deliver software on a per-iteration basis and have the customer provide input to planning and to be open for feedback.

Time-and-material contracts have the following drawbacks:

- The customer does not know whether the supplier is using time as efficiently as possible. This can lead to control mechanisms that hinder fruitful collaboration.
- The supplier has no incentive to be efficient. Being more efficient will lower profit.
- The price of the project is unbounded making it hard for the customer to budget.

Recent research in contracting shows that those contracts where risks are balanced and where all parties experience a general fairness, are most likely to create an incentive for collaboration and mutual problem-solving [2]. Refer to [3] for a comparison of many different kinds of contracts and their influence on an agile project.

### **3. Elements of the collaborative agile contract**

We want to create contracts where risk is shared fairly between customer and supplier and where likewise benefit is shared fairly. Based on our experiences we have arrived at a contract model that actually achieves this result. We call this the collaborative agile contract.

The main mechanism of the contract is to delay some of the payment until a certain criteria is fulfilled. We do not use the reaching of a calendar date as this criteria, as otherwise used in most of the contract types in [3]. Rather we want a criteria that points out a situation where the customer is getting value from the software. There is generally a mutual interest of arriving at this situation as quickly as possible. Efficiency and creativity from the supplier will be rewarded. And the customer will be careful when deciding what features are needed in order to reach that goal.

The contract defines the following elements:

- Scope described loosely in a few paragraphs as a kind of vision statement.
- An hourly price that is 10-50% below what is normal for pure time-and-material.
- A set of milestones, each of which will lead to payment of a fixed amount. For each milestone, the criteria for completion is the customer's actual deployment of the software associated with that milestone.
- A development process following agile practices.

- A suggested time frame for the overall project and for each milestone.

Note specifically the elements that are not part of this contract: the deadline is not fixed, there is no detailed requirements specification, and there are no fines. The contract establishes a space in which an agile project can take place and collaborative behaviour be cultivated.

As an example, let us show these elements from our contract with the energy corporation. [Note: We use fictional numbers, as we cannot reveal actual numbers here.]

- Scope for 6 areas of functionality described loosely by the customer on a total of two pages with illustrations.
- Development process described by the supplier: weekly iterations with requirements, estimation, prioritization, development, delivery, testing, feedback.
- Price per hour of 500 DKK, which is 50% below normal time-and-material rate.
- Delivery, split into 6 milestones, where each milestone is associated with the delivery of one area of functionality, each of which can be developed separately. Each milestone is associated with a payment amount corresponding to the relative size of the milestone against the total, as determined by a rough estimate. The total rough estimate for the 6 milestones in this case is 2400 hours and the sum of the payment amounts associated with the milestones are 1,200,000 DKK corresponding to half of the total project price if the estimate holds.
- A milestone is considered reached, when the customer deploys the software associated with that milestone. In this case the software is considered to be in production when the supplier is no longer allowed to delete data from the database.
- A suggested time frame of 9 months.

See the appendix for extracts from the actual contract.

### **4. Experiences refining the contract and working under the collaborative agile contract**

Our experience today consists of 4 different contract types with 2 customers.

Our first contract with the event bureau was close to being a fixed-price-fixed-deadline contract. However, scope was only loosely defined and the project was very small (3 weeks). The main achievement of this phase was to confirm to the customer that we had the necessary technical capability and were able to deliver quality

software on a tight schedule. This is a crucial point to their business, where new versions of the software are deployed in live events with several hundred VIP users.

When we had established a level of mutual trust, we were able to negotiate our second contract with them. This constitutes version one of our collaborative agile contract. It features a defined hourly rate, a project completion payment, a fairly detailed description of required scope, weekly iterations, an overall 3 months time frame and a price ceiling. Today we know that the price ceiling element was a mistake, and coupled with the rather detailed scope description, it created a project environment that was very similar to a fixed-scope-fixed-price project, despite the iterations and frequent deliveries. See excerpts from this contract in the appendix.

We were convinced that we could improve the contract to get a better, more collaborative process the next time. However we could not convince our customer. They preferred a standard time-and-material contract, which became the form of our third contract with them. This is rather paradoxically since this gives them more risk than a collaborative agile contract. But while we related the problems with the previous contract to the non-agile elements, fixed scope and price-ceiling, which led to more arguing than any of us liked, they concluded that time-and-material would be better. We agreed to such a contract, and at the time of writing, we have seen one of the drawbacks of pure time-and-material during the period of this third contract: In an attempt to put in a control mechanism to ensure that we used time as efficiently as possible, our customer at some point managed to force us to deliver more hours than we had capacity for. In this period we saw a clear decrease in quality and a resulting decrease in productivity.

With these experiences in mind we entered negotiations with the energy corporation. They were not at all used to agile thinking and although they liked the idea, the contract negotiations were not easy. The legal department was especially worried, which forced us to spend quite a lot of time negotiating and iterating over the wording in the contract. The notion of "full responsibility" without "fixed price" caused concern. Finally we settled on a stipulation that the supplier assumes "full responsibility" and is to be paid by the hour in fulfilling that responsibility. Finally the deal was made in the form of the collaborative agile contract outlined above. See excerpts from this contract in the appendix.

At the time of writing we are half-way through this project and have been paid two of the six milestone completion amounts. The collaboration climate on the project is probably the best any of us have experienced for the last 20 years. When the customer finds bugs or has problems with testing, setting up environments or

explaining their requirements they continuously experience that we are doing our best to be helpful. And when on the other hand we run into technical problems, find it hard to understand requirements or what the underlying business reasons are, we again and again experience our counterparts on the customer side doing their best to be helpful.

Several times the customer has experienced the benefit of the flexible-scope element of the contract, both by getting additions that they had not anticipated from the beginning and also by being able to drop some features which were not deemed valuable enough. As the supplier we have experienced how the contract motivates us to find and implement smart solutions.

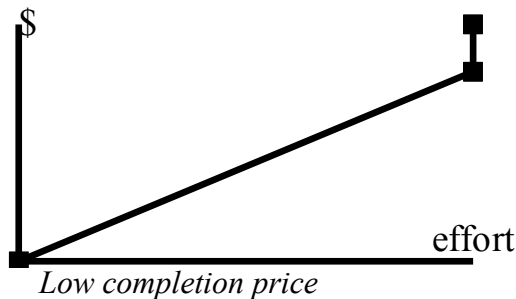
On the supplier side there has been changes in personnel, and recently one such change really showed us the value of having our development process backed up by a contract that explicitly states that the project is run as an agile project, which means minimum up-front planning compared to traditional projects. A new person on from the business sided joined the project as new product owner. Being a person used to more up-front planning, he was pushing for more detailed planning and came up with detailed draft requirements spanning several iterations. We were able to effectively push back on this request by referring to the wording in the contract. After a couple of iterations, during an analysis workshop this person suddenly uttered: "Now I understand why you delay decisions, it is not because you are sloppy, but because it is a smart thing to do".

## **5. Adjusting the Collaborative Agile contract**

An important element of contract negotiations is of course the price. In the kind of contract we are considering in this paper, there are two price elements: The price per hour and the completion price. Factors to consider when negotiating these two prices are the estimated size of the project, its value for the customer and the risks involved. Based on these estimates an estimated total price can be negotiated. In a fixed-price contract this will be the actual price. The two price elements of collaborative agile contracts make up a set of levers that can be used to tune the contract to the optimal combination of the two parameters. There are two dimensions where trade-offs can be made: the relative size of the completion price compared to the hourly price and the absolute size of the completion price.

If the completion price is small the contract becomes close to time-and-material (first illustration below). If the hourly price is small the contract becomes close to a fixed-price contract (second illustration below). It is important to

find the right balance in between these two extremes, and the balance depends on forces that will vary from project to project.



A key parameter to take into account when determining the relative balance between completion price and hourly price is the importance of fast completion to the customer. If the software is developed in an environment where time to market is crucial, then a high completion price will encourage customer and supplier to work as smart as possible towards completing fast. The underlying agile framework and hourly steady pay prevent the temptation to work in a non-sustainable way. The high completion price in this case means that being efficient pays the supplier a higher profit. On the other hand, we can have a situation where the customer wants to continue development until all features are fully in place. This can be the case when the customer is not facing a competitive situation with the software being developed. Here the supplier should go for a high hourly price, since the hourly price most likely will contribute more to total profit. This is also in the customer's interest, since the supplier will argue against extra features more vigilantly when hourly price is relatively low compared to when it amounts to a larger part of total profit.

On projects where the estimate is difficult to make, the completion price should be relatively low to ensure that this risk is shared fairly and to give the customer the required flexibility on the set of features.

On our project for the event bureau, the completion price was 10% of the total price which was probably a bit too low, given the importance of hitting deadlines.

On our project for the energy corporation, the only real drawback for us as a supplier is that the hourly pay has been too low and the completion pay too high. The current relative sizes of payment give the customer the incentive to require extra functionality before going into production because they get this extra functionality very cheap. We were convinced that the deadlines were very important for the customer, thus feeling confident that the customer would not add too many features before putting the software into production. However this turned out to be wrong, as deadlines have been pushed nearly 20%, leading to lower profit for us. To prevent that situation in the future we will be much more careful to consider accepting an hourly pay as much as 50% below normal time-and-material rate and only when we are convinced that the customer gets real value from fast time-to-market.

## 6. Future experiments

We intend to continue improving our model for collaborative agile contracts by doing more experiments with real projects. Here are some of the areas where we currently know that we have more to learn:

- Large scope changes
- Subcontractors
- Maintenance period
- Tenders
- An early exploratory phase

*Large scope changes:* During the project it might become clear that the customer needs a large new feature that was not foreseen in the original vision statement. One way to handle this could be to make a rough estimate of the new feature and negotiate an extension of the completion price based on this estimate. Then the supplier will include the new feature into the backlog of the project for normal prioritization in the next iteration.

*Subcontractors:* On the project for the energy corporation we have hired a subcontractor to fill in a gap in our capacity. We have entered a similar contract with the subcontractor based on a low price per hour and a completion price. We will later evaluate how this contract influences our collaboration with the subcontractor.

*Maintenance work:* After a period where major features have been delivered often comes a period where the customer wants deliveries with minor changes, bug fixes, and small features. We would like to improve on the standard time-and-material type contract for this type of work.

*Tenders:* Many software projects find a supplier using a tender process, sometimes regulated by law. These tenders typically require fixed-price-fixed-scope contracts. We would like to work with customers and their advisers to start experimenting with setting up tenders according to our model of collaborative agile contracts.

*An early exploratory phase:* We are convinced that many projects would gain tremendously by allocating a period of time early in the project to systematically investigate risk, establish architecture and learn about critical technical and user-facing elements of the product. Unfortunately this kind of phase fits poorly into most agile frameworks which put a strong emphasis towards creating "real" functionality from early on. Finding a way to finance such a knowledge generating period early in the project, would be beneficial for most projects.

## 7. Conclusion

Our experiences highly encourage us to continue our work with collaborative agile contracts. We have been working on projects with very constructive collaboration between supplier and customer. The customer's perceived value of the delivered software have balanced the supplier's profit. We have had experience with very different customer company types: a small, lively company and a large traditional corporation. However we still have a lot to learn. We also believe that society as a whole has a lot to learn, because we see a lot of money being wasted on large software projects that suffer under bad contracts. With our work, we hope to be able to influence and improve these practices.

## References

- [1] Mary Poppendieck and Tom Poppendieck, "Lean Software Development - an Agile Toolkit", Section on Contracts, [http://poppendieck.com/pdfs/Contracts\\_Excerpt\\_from\\_Lean\\_Software\\_Development.pdf](http://poppendieck.com/pdfs/Contracts_Excerpt_from_Lean_Software_Development.pdf), 2003.
- [2] Ernst Fehr, Alexander Klein and Klaus M. Schmidt, "Contracts, Fairness, and Incentives", [http://epub.ub.uni-muenchen.de/334/1/Contracts\\_Fairness\\_and\\_Incentives.pdf](http://epub.ub.uni-muenchen.de/334/1/Contracts_Fairness_and_Incentives.pdf), 2004.
- [3] peterstev, *10 Contracts for your next Agile Software Project*, <http://agilesoftwaredevelopment.com/blog/peterstev/10-agile-contracts>, April 29, 2009.

## 8. Appendix: Sample Contract Excerpts

Here we include excerpts from our contracts with the event bureau and the energy corporation. For legal reasons the excerpts are anonymized and numbers are fictional. The excerpts have been translated from Danish.

### 8.1. Contract with the event bureau

The contract with the event bureau was written by BestBrains and was 2 pages long:

- "The purpose of this contract is to create a foundation for an efficient and fruitful collaboration between The Event Bureau and BestBrains on the continued development of the Interactivity Software."
- "The price consists of an hourly price and a time independent bonus."
- "The Event Bureau and BestBrains use the following process to collaborate on the project: Software development is divided into two week iterations. At the start of each iteration, The Event Bureau and BestBrains agrees on priorities and accept criteria for features to develop, at the end of the iteration The Event Bureau will accept the implemented software. When software is put to use this is implicitly regarded as an accept of that software. BestBrains can invoice time spent per month. BestBrains can invoice the bonus, when The Event Bureau puts the software to use for the first time. Putting the software to use means that The Event Bureau uses the software at a customer event."
- "The Event Bureau is required to use the necessary amount of time to agree on prioritization at the beginning of iterations, give feedback as required for development during iterations, and test the delivered software for the purpose of accepting it at the end of the iterations."

### 8.2. Contract with the energy corporation

The contract with the energy corporation was written by a contract manager at the energy corporation and was 27 pages long. It was based on their standard fixed-price-fixed-scope contract and was heavily revised during the negotiations to reflect the agile contract elements:

- "The total price of the project consist of a price per hour and an incentive amount that is payed when the delivered software is put into production."
- "All software development time, including the time used to fix defects during the period of the

contract, are payed with the basic hour price until all deliveries have passed the customer's acceptance test. The supplier's responsibility implies the supplier's duty in fulfilling that responsibility as soon as possible. The time used by the supplier for this is included in the time that can be invoiced within the scope of this contract."

- "Supplier and customer have agreed that during the project, changes to the requirement specification can be made."
- "The project is divided into one week iterations with Thursday as the boundary. The development plan is maintained in a Google Spreadsheet, shared between BestBrains and The Energy Corporation."
- "The transition from one iteration to the next follows this plan: Wednesday evening BestBrains delivers software to The Energy Corporation's development environment. Thursday morning, The Energy Corporation tests the new software

focusing on new functionality and provides BestBrains with quick feedback on any defects, so that tasks from the ending iteration can be closed. The Energy Corporation can perform further testing afterwards. Thursday afternoon, The Energy Corporation and BestBrains plan the scope of the next iteration taking tasks from the backlog. The Energy Corporation ensures that the backlog is prioritized before the planning meeting. BestBrains ensures that items on the backlog are estimated before the planning meeting."

- "The customer conducts the acceptance test. The acceptance test is a test of functionality, documentation, interfaces and integration. The acceptance test is passed when there are no qualified errors. Qualified errors are errors that reduce the utility for the customer and that cannot be said to be unimportant."