Selling Agile: Target-Cost Contracts

Bruce Eckfeldt and Rex Madden, Cyrus Innovation LLC
beckfeldt@cyrusinnovation.com and rmadden@cyrusinnovation.com

John Horowitz, Esq., Grotta, Glassman & Hoffman
horowitzj@gghlaw.com

Abstract

Agile makes a lot of sense to software developers, but selling it to business people can be a tricky task, particularly in a consulting situation. While people appreciate the benefits of Agile, forming a contractual agreement that embraces change is difficult. Target-cost contracts have been suggested as a solution to structuring agile projects; but do they work? This experience report describes how we have successfully used target-cost contracting with a recent startup company. This approach improved our client relationship by ensuring that the contractual agreement empowers the development team and the client to work collaboratively to deliver software that meets the client's real needs while reducing the complexity and scope where possible. We believe these experiences will add value to teams working in both a formal consulting environment as well as in an internal development department.

1. Introduction

Agile approaches make a lot of sense to software developers, but selling Agile to business people can be a tricky task, particularly in a consulting situation. While people appreciate the benefits of Agile, forming a contractual agreement that embraces change is not simple. Target-cost contracts have been suggested as a solution to structuring agile projects; but do they work?

Over the last year, while searching for a better way to align our interests with our clients, we experimented with target-cost contracts. We found them to work well with our Agile development approach. The following case report demonstrates how we sell target-cost contracts and manage them through the development lifecycle. We then conclude with some reflections on this experience and discuss their potential relevance to similar project and client situations.

1.1. Context

We are an Agile IT consultancy and software development company based on New York City. Our services include custom software development, technology strategy, and coaching small and medium sized companies in food services, retail, and financial services. Agile Development and Lean Thinking have been core to the business since day one.

The majority of our clients are dynamic, growing companies who have quickly outgrown their technology. We typically face situations where key business systems are not working and there is extreme pressure to fix them quickly. We use lightweight technology frameworks and selectively adopt Agile techniques to fit the client and project situation.

Our typical projects are multi-release systems where each release is 8-12 weeks with 1-2 week iterations. Our project teams consist from 3-6 developers and our clients are usually off-site. We work with the client's internal graphic design teams or outside creative agency for the interface design and production; however, we maintain responsibility for the interface usability.

In addition to the two developers Bruce Eckfeldt and Rex Madden, John Horowitz, an attorney from Grotta, Glassman & Hoffman, provided input and feedback on our contractual approaches and on the development of this paper.

1.2. Motivation for target-cost contracts

Like many Agile developers, we strongly believe in the power of Agile development to create better solutions for our clients. However, we found setting up a contractual agreement that supports an Agile approach and fosters positive, productive client relationships to be a challenge.

Time and materials provides the least amount of risk for us as developers but we have found it difficult to sell because clients fear an open ended agreement and the potential for unchecked costs. Most clients' point of view is summarized well by the following quote,

> *T&M is favorable for relatively smaller budget projects or when working with a consultant in which you have a high degree of confidence. For larger projects or when working with unknown consultants we typically prefer*

fixed price, at least initially, to maintain budget certainty. Once we are confident in the consultant's ability to deliver as promised and we have a well-defined set of requirements we are more willing to shift to T&M.
-IT Manager, Asset Management Company

Fixed price is bad for us because we take on all of the scope risk and the required contingency makes projects extremely expensive. Clients also lose out in fixed price because they commit themselves to a scope too early in the project and then suffer the cost and difficulty in making changes once they realize their needs are different, what Martin Fowler calls the 'Fixed Scope Mirage' [1].

Most importantly, both approaches reinforce the traditional adversarial client-vendor relationship and limit the potential for greater success for both parties.

1.3. Informational sources

We began by looking at some of the seminal Agile and XP books for suggestions on how to set up contracts. Kent Beck and Martin Fowler have a brief 4-page chapter in *Planning Extreme Programming* that recommends a flexible scope contract and leaving the contract to merely state the number of programmers, time, and budget; leaving scope to be negotiated as you go [2]. In practice we have found this difficult to sell because clients don't want to commit to a project without some sense of what they are getting. From a client's perspective, this is like a contractor saying he's not sure how much of a house can be built for $100,000, but they'll use five people for three months, build one room at a time and see how far he can get. In *Extreme Programming Explained 2nd edition*, Beck provides a similar paragraph that suggests negotiated scope contracts as a way to provide flexibility [3].

Mary and Tom Poppendieck provide an excellent overview of several options for Agile contracts in *Lean Software Development* and include a detailed analysis of the Toyota target-cost price approach. They conclude that 'trust' is the lynchpin for these types of contracts and explain how fixed-price, time and materials, target-cost and target-schedule all help or hinder the trust building process between client and developer [4]. Unfortunately, there is little discussion of the actual mechanics of how these contracts should work.

1.4. Previous target-cost contract experience

We had two specific experiences working with a target-cost model prior to this case. We learned from both of these on how to best set up the contractual agreement and the business relationship.

Our initial attempt was actually a target-scope agreement (same idea as target-cost, except price remains fixed and scope varies). We had agreed to a fixed price contract with a client to deliver an ecommerce system in a 24 week timeframe. Then, about 10 weeks into the project, we realized that the client's expectation of the project far surpassed our understanding of the project and the defined requirements. Seeing a serious problem, we had a heart-to-heart conversation with the client and suggested we move to a target-cost contract for the rest of the project. They realized the need to repair the situation and liked the target-cost approach, however they didn't or couldn't spend more than the agreed to budget. By reducing the scope and structuring features into more granular units, we devised a target-scope agreement and left the price as defined. In this model we tracked time on each story and where we took longer than estimated, the client removed scope; conversely, when we took less time, we added scope. In order to provide incentive to both parties, days added or subtracted were discounted 50%; for example, if we finished early by 2 days, the client could only add one day of work, and conversely if we finished late by 3 days, the client only had to remove 1.5 days of features .

Our second experience was with an asset management company. The client had a spreadsheet application that needed converting and expanding into a full-fledged business application. Because there were several integration points with a legacy system and because this was our first project with this client, we suggested a target-cost model from the beginning.

It worked extremely well and the project completed within 5% of the original target-cost budget. The client was very satisfied with this approach as the quote below highlights,

It makes tremendous sense in terms of both a time value of money and as an incentive …we know that it's in DevCo's best interest to complete the project on-time or ahead of schedule in order to collect the fee sooner as well as to avoid diluting DevCo's profit margins should the project not be completed on schedule.
- IT Manager, Asset Management Company

1.5. Mechanics of our target-cost contracts

We first establish a general understanding of scope using a high-level story definition for the system. We then estimate each story at ½, 1, 2, 3, or 5 days of development for a development pair. We then add to that time for set-up, meetings, and interface design and development. This is summed to get an initial total development time. We then add a contingency percentage. This is typically between 10% for existing clients and where we have a high comfort level with the scope to 25% where the client is new and the scope vague.

The total hours (with contingency) is used to determine the target-cost for the project. We negotiate a standard daily or hourly rate based on a review of our cost structure (salaries, overhead, downtime, sales cost, etc.) and multiply this figure by the estimated hours. We then negotiate a 'profit' on the project as a percentage of the target-cost. These two sub-totals are added to establish the target price for the client.

The profit is set as a fixed amount for the project and we then bill the client actual development hours at the agreed to cost rate. See Table 1 for a sample pricing structure. The exact numbers, amounts, and billing mechanics are client and project specific, but the model is the same.

Table 1: Typical target-cost pricing structure

Line item	
Total estimated development days	50
Meetings	4
Setup	4
Interface design work	6
Day subtotal	60
Contingency/buffer (20%)	12
Total project days	72
Cost (developer/day)	$1000
Total cost	**$72,000**
Fixed profit (25%)	***$18,000***
Total estimated price	**$90,000**

From a contractual perspective, we use a two part agreement: a master services agreement to cover all non-project specific issues, and a statement of work for each project to handle project details. This structure is described in more detail in the case report.

2. Case Report

This case reports on the use of a target-cost agreement on the first release of a new project for FitCo, a start up company developing a new fitness offering. The project began in early 2005 and as of the writing of this paper the first release had been completed. The second release is scheduled to be completed by end of the summer; additional releases are planned through 2005.

2.1. The client

In late 2004, the client was in the early stages of his business planning for FitCo. Prior to this undertaking he had extensive experience in fitness and spa management and was a competitive athlete and personal trainer. He had both an undergraduate degree in sports physiology and an MBA, and for several years he had been doing operational consulting for different businesses. He had been involved in early stage investing in other companies and was familiar with start-up environments.

When we met, he had been working on his business plan and developing his fitness methodology part-time for about a year, and on a full-time basis for the last six months. The business was well thought out from a planning perspective and some initial market research was completed, but the specific methodology mechanics were only defined at a high level.

Key to the ultimate success of this project was that this client was not risk adverse. He was comfortable taking calculated risk and used to managing risk as part of the business. He understood that risk was not only an inherent part of starting a company, but that risk-taking was crucial to being successful as a start-up.

2.2. The project

The exact details of the offering or workings of the FitCo system were very vague at the beginning of the project. Having only several PowerPoint presentations and excel spreadsheet of training plans, there were very few exact details to determine project scope. The business plan called for an online and physical location offering and the system needed to support both environments. There were two components: 1) an underlying business logic and customer management framework to structure the fitness planning process and 2) a sales and marketing system to promote and entice people to become members.

The initial objective was to get a version of the system running with a group of alpha users so FitCo could prove the methodology and establish a basic revenue stream. At this point, the client would begin looking for early stage investors using the system as a demonstration of the plan. He also retained a marketing firm, architect, and product designer to develop other parts of the business plan in order to create a compelling investment opportunity.

2.3. The sales process

We started with a two-hour meeting with the client where he explained his business vision and we captured high-level stories for the purposes of generally estimating the project. See Table 2 for example stories that were captured at this point.

Table 2: Example High-Level Stories

Story	**Est. days**
Printable exercise diagram	2
Unlocking workout content	1
User can print out workout	2
Add image/animation to exercises	3

Within a few days the stories were reviewed with the client. We then had him sort the stories into three groups: 1) alpha/proof of concept, 2) beta version/pre-investors, and 3) full market version/post-investors. Each phase was approximately eight weeks of development time once we added set-up time, meetings, and a reasonable contingency to accommodate the expected increase in scope.

The client was funding the initial development with his own money so he was extremely cost conscious. However, he also understood the importance of having a high quality software system. We both agreed that the scope was highly variable; but the client didn't like a time and materials arrangement and we certainly weren't going to fix price a project with so many scope questions.

We presented the target-cost model as a way to align our incentives and work together to develop the best system in the shortest amount of time. The client liked this model for three reasons: 1) putting our profit at risk made him feel that at some level we were "in it" with him, 2) the client felt like that he had the opportunity to reduce cost by finding ways to simplify the complexity without sacrificing quality, and 3) he felt like we had a clear disincentive to stretch the project out to increase our billable time. His perspective is nicely summarized by the following quote from our client,

Most importantly, it puts my team and the development team on the same goal path, which has created a "we're in this together atmosphere." As a startup business owner - that's critical.

- Owner, FitCo

With all of the decision makers in the room and significant pressure to start work quickly, the entire meeting only took three hours; the result was a letter of agreement (LOA) which included scope for the current release, the target-cost and billing rate, and our fixed profit for the first phase of the project.

2.4. The agreement and contract

Based on previous experience working with clients under time pressure, we used a LOA to start work and in a parallel process started on a master services agreement (MSA) to cover the remaining contractual issues. Our general contracting strategy is to use a LOA to cover project related details and a MSA to cover all non-project specific information regarding the terms of the relationship. This typically includes defining confidential information and non-disclosure terms, ownership of work product, indemnification, any non-compete agreements, and several specific legal sections related to the administration of contracts. Occasionally we'll define standard rates/prices and terms for their increase over the life of the relationship and a section on terminating a work order and early termination fees; however we typically don't include these since they are often project specific.

Typically a MSA takes from one to eight weeks to finalize, depending on the type of client organization. In this case the MSA took approximate four weeks; and as is usually the case, the majority of the time was spent discussing just a few issues. Although some people have suggested that starting a project without having a MSA signed can be risky, we typically find once the project starts, the client is under more pressure to finalize this than we are, so having the project underway works to our advantage.

2.5. The development process

The development proceeded like our many other Agile projects. The entire team met with the client to identify the most important features and spent about two hours sketching out models and interfaces of how the system would work. Fortunately, the client was on-site so we didn't need to review every detail knowing that we could take advantage of immediate feedback and answers to questions.

Based on the LOA we had assigned an estimated number of development days to each story. We then converted these days into development hours and entered them into our home-grown time tracking system we developed for target-cost contracts. This system allows us to track work against stories and monitor our actual time spent against our original estimates. This tool allows us, and the client, to see how development is going at a story level and monitor overall progress.

Table 3: Sample of Hours Tracking System

Story	Act.	Est.	Net
Printable exercise diagram	12	32	-20
Unlocking exercise content	18	16	+2
User can print out workout	31	32	-1

We reviewed the overall hour status with the client at the end of every iteration at a minimum; often this was done weekly. We looked both at individual story hours as well as overall project hours. The project hour summary became such a central tool to managing the project that both the team and the client would often ask to see the summary when potential changes to the system were being discussed in order to manage scope.

In this way the target-cost contract did exactly what it was suppose to do: make each side consider the time and budget impact of scope decisions and balance the need vs. the cost.

2.6. Adding to the scope

Increasing the scope to the project fell into one of three categories: fixes, clarification, and enhancements. **Fixes** are changes to meet the agreed upon functional requirements implied by the existing stories. For example, we agreed to a feature that calculated the total time of several sub-components of an exercise program. However once in the project we realized we hadn't provided for the ability to enter the times at the sub-component level. This was considered a fix.

Changes resulting from customer feedback were **clarifications**. At one point we realized that a certain pull-down list really need to include more than what was specified in order to work correctly. It wasn't clear at the time of scoping, but once the client started using the system, it quickly became apparent that the system wouldn't work correctly without changing this.

New stories that need to be added to the system were **enhancements**. When developing an exercise editor, the client decided he wanted to be able to assign tools to exercises (i.e. barbell, dumbbell, Swiss ball, etc.). Because we could effectively implement this change as a separate story and the rest of the system could function without it we classified this as an enhancement rather than clarification.

For fixes and clarifications, we merely billed additional hours against the associated story. If we exceed the estimate, the additional time was covered from a cost point view; however, we didn't make any additional profit. Conversely, an enhancement was treated as an increase in the scope of the project and therefore justified an increase in profit. In order to calculate the profit increase, we estimated the change in development days, added contingency and any required setup time in order to calculate a revised target-cost of the project. We then adjusted the fixed profit based on this revised target and billed the different. (See Table 4 for example.)

Table 4: Calculation of a Type 3 Change

Line item	
Total estimated development days	50
Additional development days	*+ 5*
Meetings	4
Additional meeting time	*+ 1*
Setup	4
Interface design work	6
Revised day subtotal	66
Contingency/buffer (20%)	13.2
Total project days	79.2
Cost (developer/day)	$1000
Revised Total cost	$79,200
Revised Fixed profit (25%)	$19,800
Original fixed profit	$18,000
Additional fixed profit to be billed	**$1,800**

3. Reflections

The following are reflections on our experience from this case and our previous target-cost contracts. Our hope is that others can learn from them and apply them to their projects to improve the development team's relationship with the business owners.

3.1. Project size

This case is on the smaller end of the gamut of project sizes; however we have extensive experience with larger projects in the 15-20 million dollar scale involving teams of 20-30 developers for many months and we feel these experiences can be scaled to these types of situations. The problems we've seen on smaller projects (misunderstood scope, managing feature creep, meeting development deadlines, etc.) are many of the same problems we've experienced on larger projects.

Smaller projects are also more sensitive to variance and risk. Because we don't have long development cycles to average out feature variability we need to be especially careful to limit our exposure. A large project can absorb an extra week of work over a ten month schedule; smaller projects cannot. Therefore in this case we needed to be even more vigilant about managing the scope and our delivery of value to the client than a larger project.

In practice, we have found that big projects are really lots of smaller projects. We often find ourselves unraveling plans to find these more manageable morsels that fit our target-cost model well. We're not sure if this is in spite of our experience with large projects, or because of it, but it seems to happen.

3.2. Buffer

Being avid readers and proponents of Theory of Contraints, we have been using the approach suggested in Eli Goldratt's *Critical Chain* for handling buffers on the project [5]. We removed 'padding' from our stories along with allocations for meetings and setup time and place them separately in our estimates. We then placed a reasonable amount of extra time within each iteration to provide slack in the schedule to accommodate variance and change.

On the project we identified and tracked our time in four different areas: 1) fixed complexity that could have been identified during initial scoping, 2) time required for change based on client feedback, 3) meetings and design sessions, and 4) deployments. We are now using these measurements as guidelines for determining estimates in new projects.

3.3. Risk tolerance

We don't believe that this project would have been as successful had the client been more risk adverse. In this case the client was comfortable with risk and understood how to calculate it to his business advantage. He understood there was a level of variability in the project and was willing to work in an agreement and development approach that embraced change. Our prior target-cost experience with an asset management company, another client that understood variance and calculated risk, was also extremely positive.

Retail clients have been less willing to do target cost contracts. We believe that this is because of the nature of the retail business and their comfort with risk. Retail margins are very tight, and although managers can see the potential gains in a target-cost approach, they prefer the risk mitigation of fixed-price even at the expense of reduced potential benefit.

3.4. Project and iteration size

We generally structure our releases to be approximately 8-12 weeks in length (although we have scoped projects as long as one year). We have found that releases longer than that typically have a significant amount of feature change. Target-cost contracts that have extensive feature change are cumbersome to manage due to the required tracking of new and old features.

Because we structured the project to be in several smaller releases, we had only two new features added and none removed within the first release. The number of changes would have been higher had we made the release bigger. Several of the features originally planned for release two had already been canceled or dramatically changed prior to the end of the first release. If we had made the first release 16-24 weeks (something the client originally wanted to do) these [feature changes] would have had to have been removed or dramatically re-scoped, making the tracking more involved and time consuming for the team and client. By keeping the releases short, we were able to make clean breaks between releases and avoid excessive time tracking scope changes.

3.5. Client relationship

One of the great benefits of this project, and the reason we continue to support the use of target-cost contracts, is the positive relationship we are able to build with our clients.

By having "skin in the game", we are perceived as being a business partner rather than just a software vendor. Clients invest large sums of money to build these systems in hopes of gaining significant business benefit at a later time. Target-cost contracts allow them to share some of the risk that the development will take longer than estimated with us.

This shared risk also creates an incentive for the client to think of ways to reduce cost and complexity. If meeting times can be cut in half, the client will endure the trouble of being on-site to answer questions and work with the development team. If providing an internal developer will speed up development, they'll make the person available. Conversely, if the internal developer will hinder the team, they will think twice about insisting on it.

By aligning our business interests we're eliminating the typical win-lose environments of most development projects and as a result, we are both aligned to reduce complexity and cost on the project.

3.6. Tracking time in target-cost

Time tracking and reporting becomes an important issue in target-cost contracts. Traditional fixed price contracts don't typically have the pressure to track hours since the total development time doesn't have a bearing on the project price. Time and materials contracts obviously need time tracking, but only for the purpose of billing the client at the end of the project/billing cycle; time is not 'watched' during the project (in fact, some would argue that actual time spent is strategically obscured until after the project is completed so as not to alert a client of overages until it is too late to do anything about them).

In target-cost contracts, we've found it key to have a good time tracking system that records time on a story by story basis. By tracking time by story, the team and client can clearly see where development hours are going. Time reports can quickly identify features that have become unnecessarily complex and provide the client with a real-time status of the project outlook.

3.7. Business transparency

In negotiating target-cost contracts, we've found it necessary to be fairly open with our company finances in order to justify our cost rates. We typically use actual numbers (copies of rent agreements, etc.) or publicly available industry information (salary rates from job postings and industry reports, etc.). A malicious client can use these to negotiate against you, however many clients have found this transparency refreshing and an overture towards building a relationship based on trust.

3.8. Handling changes on a target-cost project

Unfortunately target-cost contracts remain about as complex as a fixed price agreement with regard to changes to scope. Two specific complexities presented themselves in the case: 1) deciding whether a change was just an unanticipated complexity of an in-scope feature (the extra time billed at cost) or was it a new feature (therefore, we should increase our fixed profit) and 2) if the change was determined to be a new feature, figuring out the appropriate increase in fixed profit.

Determining the nature of a change was a bit of a negotiation, although a much less dramatic one than our previous experience on fixed-price contracts since only our profit is at risk. Once we agreed with the client that a feature was indeed a 'new' feature, we had to determine the increase in the fixed fee. We effectively had to recalculate the entire revised scope of the project, figure the new fixed fee and bill the difference. In the future we plan on handling this by establishing a profit multiplier as the fixed profit divided by the development days subtotal, which would cover contingency, meetings, etc. For example, using the data in Table 1, the development days subtotal is 60 days and the fixed profit is $18,000, therefore the multiplier in this case would be 300. Then when the client added a story that took say 3 days, we would just increase the fixed profit by $900.

On the flip side, we haven't had to remove scope that wasn't traded for something else. Theoretically, we could find ourselves in the situation where the client wanted to remove scope to the point where our total estimated development days is less than what we originally estimated. The issue is whether we would then reduce the fixed fee as well. Also, given our need to commit resources and schedule other projects, a reduced scope below original estimates would leave us with a gap in our staffing schedule (a very bad thing in the consulting business). The most obvious option would be to set the original cost estimate and fixed fees as minimums to the contract; however our success of selling these terms has yet to be proven.

3.9. Applying learnings to internal teams

Although internal teams don't have the formal contracts with the client that consulting groups do, there always exists an informal contract between the development team and the business groups; and many of the same challenges over scope and deadlines present themselves. Teams working with internal groups can still measure target and actual time and use the same target-cost approach to create incentives for both developers and business teams to reduce complexity and shorten development times.

Vacation time or financial bonuses can be used to provide developers with incentive; internal daily or hourly rates can be used to provide incentive to business units. If a team can finish early, the business unit gets charged for less time (or has the option of adding more features) and developers accrue vacation or bonus early and potentially have the opportunity to increase their total yearly compensation.

Even though the formality is not the same, the mechanics can be used effectively. With a little creativity, teams can adopt the model to fit their specific cultural and corporate environments.

4. Conclusion

Traditional time and materials and fixed price contracts don't work in most Agile projects; they fail to embrace change and provide the flexibility an Agile team needs to create the best system for a customer.

As this experience report shows, target-cost contracts offer a unique and promising alternative that allows developers to share risk with clients and for clients to feel like developers are working with them to make the project successful. As this last quote shows, a target-cost approach can make a world of difference in creating a positive, Agile-friendly business relationship.

The target-cost contract model has worked extremely well for my startup venture because it incentives efficient, yet flexible, software development
- Business Owner, FitCo

5. References

[1] Fowler, Martin, "Fixed Scope Mirage", September 30th, 2004 blog entry on http://www.martinfowler.com

[2] Beck, Kent and Fowler, Martin, *Planning Extreme Programming*, Addison-Wesley, New Jersey, 2001, pp. 121-124

[3] Beck, Kent, *Extreme Programming Explained*, Addison-Wesley, Boston, 2004, p 69.

[4] Mary Poppendieck and Tom Poppendieck, *Lean Software Development*, Addison-Wesley, New Jersey, 2003, pp 161-177.

[5] Goldratt, Eli, *Critical Chain*, Great Barrington, 1997.