



Update

Contracting Agile Projects

by Jens Coldewey,
Senior Consultant,
Cutter Consortium

After more than five years of agile software development, we understand its nuts and bolts pretty well: we know how to build software in an agile manner; we know how to set up an agile team; we have built an impressive set of powerful support tools; we know the caveats and limitations. And a year ago, the Standish Group named agile development as a major success factor in a software project.

This sounds like good news. However, from a business point of view, this is only half the story. Agile development thrives among the fast decisions from close communication between a still somewhat blurred “customer” and the development team. Together, customer and development concoct what the system looks like and compass the project following what they understand to be the business objective.

This works well in startups and small companies, but large organizations run into severe problems: agile foils most controlling systems and makes it difficult to define the deliverables, the basis of a contract. This creates a delicate situation: agile helps to fulfill a contract, but a contract seems to need a basis that is far away from the agile idea. One of the major benefits of agile is that you can *quickly change* circumstances, if your circumstances demand flexibility. One of the major benefits of delivery contracts is that

they don't allow the other party to alter what is delivered just for their good. One of the major drawbacks of contracts is that they don't allow you to change either. Thus, a standard software delivery contract is contradictory to agile.

This is not special to agile, but to most software development efforts in general. Attempts to specify a system “legally safe” fail regularly, such as the FISKUS project conducted by the German tax authorities: the project was stopped after 10 years with virtually no productive software running. The teams gave up their attempt to specify a tax-processing system for all 16 German states — fast changes to the legal aspect was only one part of the problem.

To make things even worse, there are project crashes *because* of the contract. I have seen projects, where everyone knew that the contracted system did not work, but no one had the guts to renegotiate the carefully balanced contract. In one project, the technical staff dealt with enormous efforts to deal with the problem within the boundaries set by the contract but missed budget and deadline. In a second attempt to complete the project, management focused on delivery instead of fulfilling the contract; the project finished successfully two months before its schedule.

So on one hand, we need contracts to secure a fair and professional business relationship; on the other hand, poorly constructed contracts have the potential to nullify any business objective the project has. A careful contract design is therefore necessary to ensure that contracts help in running a successful project as well as to ensure that close customer collaboration is more important than contract negotiation — so the project can fulfill your business objectives instead of your lawyer's.

LAWYERS AND AGILE CONTRACTS

Lawyers are adept at writing contracts that offer their clients maximum advantage during a lawsuit. However, they are not experts in what to write a contract about. What you need is a contract that maximizes the efficiency of software development and thus minimizes the risk of a lawsuit occurring.

Recently, I attended a talk by an excellent lawyer on software contracts. Her message was quite clear: write a sound and detailed specification and contract the vendor based on this document. Seems simple enough, but I raised my hand and asked her how to deal with the fact that projects are much more efficient and have fewer risks when done iteratively and in an agile way as compared to the waterfall style she had laid out. “Well, these are business problems, and I'm not a business expert” was her reply.

She was right: the life of a lawyer would be much easier, if perfect specifications were written right from the beginning — and the life of a project manager would be easier, too. But project managers have tried this path for 20 years without any significant success. And some “experts” claim that the method we use is not right, and we only need

to use/buy their great tool or process to solve any problems. But they have been telling us this for 20 years, and their claims resemble Jerry Weinberg's first law of bad management: “If what you're doing isn't working, do more of it” [3].

The agile community has carefully worked out why the waterfall approach doesn't work and has developed alternative techniques that do. These techniques include not to design everything up front. So the advice the lawyer gave during her talk not only maximizes your chances to win a lawsuit, but it also maximizes the chances that you have to face one.

So don't consult your lawyer too early. To contract an agile project, you must identify what to contract first and then consult your lawyer to turn that into legal speak.

THE OBJECTIVES OF AN AGILE CONTRACT

In her book *Lean Software Development*, Cutter Senior Consultant Mary Poppendieck proposes to contract the project objectives rather than the solution [2]. She proposes different models used in manufacturing (e.g., target-cost, target-schedule, or shared-benefit contracts). The basic idea is to grant the vendor a financial incentive if the project objectives are achieved — and a penalty if they aren't. Observe that these goals are business objectives rather than technical objectives, such as implementing a certain specification within a given time frame.

I know several software firms that offer these constructs to their clients — but I have never seen a client accept such an offer. There are two problems. First, from a practical point of view, the true objectives of a project are difficult to transfer into money unambiguously — a precondition of turning them into a court-safe contract. The

second problem is more fundamental. A contract should distribute responsibility fairly so that individuals are only responsible for what they can influence. Shared-profit contracts replace this through a fellowship of fate, which is not very popular to purchasing, legal, and controlling departments.

To find a way out of this maze, let's recall what is usually known at the beginning of an agile project:

- **The business objectives you want to achieve together with some ideas on how software can help.** These objectives include the time and resource budgets that you are willing to spend.
- **A well-defined planning and acceptance process for each iteration.** This includes iteration interval and meetings (e.g., planning game, sprint planning meeting).

This is what both parties agree upon, and this is what an agile contract should contain. Both parties agree to run a series of iterations, to plan and implement each iteration according to the state-of-the-art, and to accept them in reasonable time. With minimal additional effort, you can make these steps traceable to a level that at least matches the level of waterfall methods.

Since state-of-the-art is quite a high standard in agile development, your chances to get what you pay for are better than with a simple fixed-price contract as are your chances to achieve the effort's objectives efficiently.

So instead of contracting a result, you contract a process that leads to this result. The process must provide just enough formality to make agreements safe for both sides: for every iteration planning session, there should be written minutes both parties agree upon.

Acceptance should be based upon automated tests, and these tests should be documented. This adds a few pages per month on extra documentation — additional “ceremony” that helps you and your vendor remain partners, although there is money at stake.

DON'T SINK THE SHIP — STANDARD CONTRACTS FOR AGILE DEVELOPMENT

From the client's point of view, agile development consists of two parts. The first is a learning and setup phase. The development team learns about your business, your objectives, and your way of thinking. You learn about technical options to achieve these objectives and about iteration strategies and limitations. In the end, both parties have a common vision about what the system looks like, both from a business and technical perspective. This is not a six-month “inception phase” but a short period of time between three and six weeks. If you have worked together with this particular development team before, it may even shrink down to a single three-day workshop.

Observe that the objective during this part is to learn and understand, not to deliver some “specification.” This means you have a classical case of a service contract, similar to a coaching contract.

At the end of the setup phase, you are ready to start the second part: a series of short iterations. At the beginning of each iteration, there is a process on how to define the functionality of the next iteration (the planning game, speculation, or sprint planning meeting). At the end of these meetings, you come up with the following:

- List of mandatory functionality for the coming iteration

- Estimation of the development effort for each item on the list

- Consent between you (the client) and the development team that this iteration makes sense both from a business and technical perspective

Formally speaking, this is the basis you need for a standard fixed-price contract: you define what to build (your list) in which time frame (the next iteration) and how much this will cost (the estimation). In practice, this is even a better basis than a traditional setting because the estimations are usually much more accurate in a steady agile project.

So the whole project consists of a service contract followed by a series of fixed-price contracts as long as working on the software pays for the benefit. In Extreme Programming (XP), these are called “negotiated scope contracts” — standard contracts that capture the essence of how an agile project runs.

Traditional methods have also used this split between a consulting part and a delivery part. However, these two parts were usually done by different teams and on a different scale. In agile, the development team collaborates with the client from the very first second.

It is not a good idea, of course, to start negotiations every iteration anew. A framing contract fixing this process — and the conditions of both phases as well as the overall budget — makes sense. With each iteration, a little bit more content is filled into this scaffold.

At first glance, the idea of an infinite series of contracts conflicts with the need to set up an overall budget for an effort up front. The deeper reason here is that agile does not necessarily distinguish between the development and maintenance phases of an effort.

Rather, you try to enter a smooth maintenance mode as soon as possible because maintenance means that you focus on the running software to make money rather than on a future promise to make money, once the software is eventually complete.

If you are — like most managers — unhappy with a short budgeted period of development and a long, open-budget phase of incremental improvement, there is no problem with prolonging this first phase. For example, you could set up a budget of 12 iterations and a list of features the domain experts think are minimum to deliver business success. If you leave enough slack to react to new findings, you can still be “agile enough,” but beware: this contract gradually leaves the ground of agile development and should be used as a delicate balance between the two business needs of planning and agility.

In large organizations, there is another caveat with this approach. Since there is a new contract for each iteration, the purchase department, for example, may suggest something like, “We have this company on the other side of the world that offered to fulfill the same contracts for 10% less. So contract them for development and reduce the current company to write these contracts.” Don't agree to this! Agile development thrives on close collaboration and on the fact that those who suggest solutions also have to build them, learning more and more about what works well and what doesn't. To change the horse during the race means you have to pay much more than you can ever save. In horse races, the same rider runs many races with the same horse for years because it's best when both horseman and horse are jelled. Likewise, agile development works best when client and vendor are well-jelled.

MORE THAN A PROJECT — MANAGED RELATIONSHIPS WITH AGILE VENDORS

To see the complete picture, it's necessary to take one step back. Agile development is about building mutual trust between the business experts and the software people. If you contract development to another company, this maps to mutual trust between your organization and the vendor. If you talk about trust in business, you sometimes get reactions as if trust was something from times of yore when people were still nice to each other. But trust is nothing more than a strategic partnership, and the way to get there is through vendor management.

Most large organizations maintain a vendor profiling system to manage the relationships with their vendors. This database contains information about skills, rates, and former projects that have been run with each vendor. More or less, sophisticated ranking systems categorize the vendors from strategic partners to vendors the company wants to get rid of. Used wisely, these systems can establish both a business and a legal relationship with your vendors in which you don't have to step on the thin ice of contracting a single agile project.

Unfortunately, many companies confuse "efficient" with "cheap" and use vendor management only as a way to put vendors under pressure (e.g., "reduce your rates by 40%, or we'll remove you from our system"). These techniques may reduce their average rate by 30%, but chances are you're left with vendors that have little chance in the free market and cause more harm than good.

The problem here is that you trade numbers that are easy to measure with features that are difficult to quantify. You trade rates with efficiency and subordination

in contract negotiations with competency. The default is that the focus is on the numbers that are easy to measure, while the other variables erode. An executive once told me, "All project managers have protested against this decision, but they block saving attempts anyhow, so we do not care." It was this very decision that had led to the project crashes I was called in to fix.

Some purchase departments have reacted to these problems and rank their vendors based on several criteria such as rates, size, and evaluations from previous projects. This is a wise step — as long as these rankings reflect the reality of software development. As Cutter Fellows Tom DeMarco and Tim Lister pointed out in the 1980s, productivity ranges by a factor of 20 between individuals in software development [1]. If you exchange a vendor with another that has only a fourth of the productivity, it is a bad deal, even if their rates are 40% lower, and they are much nicer in negotiating these rates. Therefore, rather than focusing on vendor process certifications, size, and rates, the ranking should focus on criteria such as the following:

- Productivity
- System quality
- Flexibility
- Collaboration level

Observe that none of these criteria are easy to measure. You have to rely on the assessment of the managers who collaborate with the vendors. On the other hand, this vendor evaluation scheme allows you to find long-term partners that understand your business and help you to align your IT projects with your business objectives. Working — and contracting — with vendor partners can help reduce your costs and, at the same time, improve the value of your IT.

SUMMARY

Many organizations hesitate to contract agile development efforts because agile contradicts a fixed-price approach. Insisting on a fixed-price contract would mean to give up on the efficiency and competitive advantage that agile offers to the business department. Completely new contract forms have been supposed but are not broadly accepted due to the legal problems connected to them.

The easiest way to contract an agile project is to set a lightweight contract with each iteration, based on a framing contract about the collaboration. This uses standard legal constructs and still balances flexibility, security, and efficiency. You can even extend these contracts to cover several increments to deal with fixed budgets, but this is a dangerous game: the risk being that you fall back into not-so-good old waterfall practices.

Beyond a single effort, agile development is about building trust. A special vendor management database may help large organizations here — as long as the vendor evaluation is based on "soft" criteria, such as collaboration level and software quality, rather than on "hard" facts, such as rates and willingness to subordinate during negotiations.

REFERENCES

1. DeMarco, Tom, and Tim Lister. *Peopleware: Productive Projects and Teams*. Dorset House, 1987.
2. Poppendieck, Mary, and Tom Poppendieck. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003.
3. Weinberg, Gerald. *Quality Software Management I: Systems Thinking*. Dorset House, 1992.

ABOUT THE AUTHOR

Jens Coldewey, based in Munich, Germany, is a Senior Consultant with Cutter Consortium's Agile Project Management practice. He specializes in deploying agile development and object-oriented (OO) techniques in large organizations. Mr. Coldewey has introduced agile practices in numerous business projects for clients such as Deutsche Telekom, Siemens, Dresdner Bank, the German Insurance Association, Generali Group, Zurich Financial Services, and Swissair Group. He is a founding member of the Agile Alliance and a regular speaker at agile and OO conferences. Mr. Coldewey's publications include "Relational Database Access Layer" in Pattern Languages of Programming Design 3 (edited by Robert Martin, Dirk Riehle, Frank Buschmann) and "An Access Layer for Object Databases" in Object Databases in Practice (edited by Mary Loomis and Akmal B. Chaundhri). He can be reached at jcoldewey@cutter.com

The *Executive Update* is a publication of the Agile Project Management Advisory Service. ©2006 by Cutter Consortium. All rights reserved. Unauthorized reproduction in any form, including photocopying, faxing, image scanning, and downloading electronic copies, is against the law. For information about reprints and/or back issues of Cutter Consortium publications, call +1 781 648 8700 or e-mail service@cutter.com.

Workshop Developers/ Presenters

Every workshop is led by one of Cutter Consortium's expert Senior Consultants — experienced IT professionals who have honed their skills and developed their methodologies over years in the field, at companies like yours.

Verna Allee
 Scott Ambler
 Rob Austin
 Christopher Avery
 Sam Bayer
 Kent Beck
 E.M. Bennatan
 Bob Benson
 Tom Bugnitz
 Ken Collier
 Rachel Davies
 Tom DeMarco
 Jonathan G. Geiger
 Sid Henkin
 Jim Highsmith
 Wendell Jones
 Jeff Kaplan
 Joshua Kerievsky
 Bartoz Kiepuszewski
 Tim Lister
 Lisa Loftis
 Michael Mah
 Terry Merriman
 Larissa Moss
 Ken Orr
 Carl Pritchard
 Ken Rau
 Thomas Redman
 Suzanne Robertson
 Mike Rosen
 Michael Schmitz
 Rob Thomsett
 Jim Watson
 Karl Wiig
 Bob Wysocki

Cutter Consortium
 37 Broadway, Suite 1
 Arlington, MA 02474-5552, USA

Tel: +1 781 648 8700
 Fax: +1 781 648 1950
 Web: www.cutter.com
 E-mail: sales@cutter.com



Workshops

In these times of intense pressure to make every development dollar and every development minute count, the maxim *you are only as strong as your weakest link* has never rung truer.

Moving your development organization up the productivity curve will improve the ROI of every one of your projects. Just trace this back and you'll discover the ROI in training is immense. And with training and workshops designed and delivered by Cutter Consortium's Senior Consultants, you can add to that equation the peace of mind you get from being trained by the best of the best.

Cutter Consortium offers inhouse training solutions from IT project management techniques to software development methodologies, improving data quality, architecting Web services applications, aligning business and IT objectives, and more.

●●● Workshop Topics

- Agile Development Methodologies
- Agile Project Management
- Business-IT Alignment
- CIO Dashboards
- Data Quality
- Data Warehousing
- Enterprise Architecture
- Estimation Techniques
- Extreme Programming
- IT Strategic Planning
- Knowledge Management
- Metrics/Benchmarking
- Outsourcing
- Requirements Management
- Risk Management
- Software Development Practices
- Teamwork and Leadership
- Web Services

For details about the courses offered in each of these areas, contact Dennis Crowley at +1 781 641 5125 or dcrowley@cutter.com, or visit www.cutter.com.