adVANTAGE: A Fair Pricing Model for Agile Software Development Contracting

Matthias Book[1], Volker Gruhn[1], and Rüdiger Striemer[2]

[1] paluno – The Ruhr Institute for Software Technology, University of Duisburg-Essen
Gerlingstr. 16, 45127 Essen, Germany
{matthias.book,volker.gruhn}@paluno.uni-due.de
[2] adesso AG, Rotherstr. 19, 10245 Berlin, Germany
striemer@adesso.de

Abstract. Agile software development methods are harder to adopt by third-party software developers than by in-house software development teams, since traditional contractual frameworks can easily lead to unfair risk distributions between client and supplier when applied to agile projects. We therefore present a pricing model for agile software projects that distributes risks evenly between the partners, and encourages efficient, high-quality contributions on both sides.

Keywords: Agile processes, third-party developers, contracting, pricing.

1 Introduction

The development of complex information systems is typically subject to a number of common challenges – overrunning time and budget constraints, missing requirements and quality expectations, and dealing with continually evolving business environments and integration landscapes. As Curtis et al. already described almost 25 years ago, these problems mostly stem from insufficient communication between business and technology experts, or users and developers [1].

These challenges occur in in-house development projects, where a large company's IT department develops its own systems, as well as in external development projects where the software is developed by an outside contractor. Each project situation brings its own set of communication and collaboration challenges that contribute to the typical problems that complex software projects often face. In this paper, we will focus on one aspect that makes external software development even more conflict-prone, namely the contracting and pricing details of such projects.

1.1 Challenges in Traditional Project Contracting

In the experience of IT service providers who develop software solutions tailored to clients' individual requirements (e.g. building custom information systems for insurances or healthcare providers), most clients initially have only a rather coarse, high-level idea of the system they need. At the same time, however, they would like to negotiate a fixed price or budget ceiling for the project.

C. Wohlin (Ed.): XP 2012, LNBIP 111, pp. 193–200, 2012.

In theory, such a fixed price or ceiling could only be properly established based on a complete specification of the system to be built. However, creating such a complete specification typically is neither economical (since it requires considerable effort on both sides, before the actual project even begins), nor is it helpful (since the client typically remains unable to express all his requirements in sufficiently complete and consistent detail up front).

In practice, faced with the client's insistence on a fixed price despite the lack of a proper specification, service providers often end up trying to make a best guess of the price, trying to balance the expected project effort with the aim to under-bid any competing providers. Since this usually results in too low of a price, the contract will typically be won by a service provider who subsequently struggles with his too-low bid, or who is an expert in playing the inevitably ensuing change request game. Obviously, neither constellation is conducive to a lasting customer relationship.

1.2 Challenges in Agile Project Contracting

The recognized lack of helpfulness of complete up-front system specifications has led to the rise of agile development methods such as Scrum [2], where voluminous specifications are replaced by quick iteration cycles. At first sight, the continuous refinement of prototypes in tight collaboration of users and developers seems like an ideal solution to the difficulty of specifying a complete system beforehand. While the model lends itself naturally to in-house projects, clients and service providers have however found it hard to transfer to the commercial domain:

For an agile project, it is virtually impossible to set a fixed price – since the project scope and the required solutions materialize only gradually, and prototyping implies performing a considerable amount of work that does not make it into the final project, but is discarded, the actual effort is hardly foreseeable. A fixed-price contract would thus expose the service provider to the complete project risk, while the client is tempted to keep adding bells and whistles along the way at no additional cost.

On the other hand, running agile projects on pure time and materials (T&M) contracts is equally undesirable: While these seem more fair at first sight (since the payment corresponds exactly to the work done), they actually incentivize service providers to blow up the development effort and neglect quality control at the client's expense. The project risk thus lies fully with the client.

Again, neither situation is satisfactory for both parties. It would therefore be desirable to find a contracting model that has a built-in risk limitation mechanism for the service provider, and a built-in cost limitation mechanism for the client. In this paper, we will propose such a contracting model that ensures fair distribution of risks and encourages efficient work on both sides, without relaxing any of the contractual obligations (e.g. warranty) that parties in a large-scale development project would normally also expect from each other.

In the following section, we will review related works that we employ as building blocks of our approach. In Sect. 3, we will then introduce our agile contracting model adVANTAGE, and discuss initial practical experiences with it in Sect. 4.

2 Related Work

To improve the "thin spread of application domain knowledge" [1] and the inter-stakeholder communication issues that are at the heart of many troubled software engineering projects, numerous approaches, process models and tools have been introduced over the years. We believe it is necessary to find a pragmatic mix of these methods and tools for each project, since the solution often lies less in thorough formalization of a system than in solid understanding of its relevant aspects.

In our approach, we strive to reflect this conviction not just by choosing an agile process model, but also by supporting and encouraging it in the software project's contractual framework. The adVANTAGE approach is therefore grounded in two software project management concepts:

To focus on the relevant aspects of a system, Boehm and Huang introduced the notion of value-based software engineering [3] that integrates consideration of a feature's or component's value within a system into all software engineering activities. As detailed below, our contracting model enforces value orientation throughout the project's progress by attaching price tags to each user story.

As Lehman pointed out, stakeholders in every software project are bound to experience a considerable amount of uncertainty in their specifications and decisions [4]. Just as software engineering methods and tools should acknowledge these uncertainties by guiding stakeholders to resolve open questions, however not force them to fix answers prematurely, our contractual framework is designed to leave room for gradual elimination of initial uncertainty by allowing re-prioritization of the project deliverables and late detailing of user stories as the project progresses.

Beside the software engineering aspects, there is also a variety of suggestions for shaping the business and legal aspects of agile relationships: Van Cauwenberghe proposed a number of guidelines for reducing the risk in agile fixed price contracts [5]. As we will see below, adVANTAGE employs some similar mechanisms (e.g. letting the customer prioritize), however differs significantly in that it is not a fixed-price model. Sutherland also suggests mechanisms for risk sharing and ensuring sustained mutual involvement in his "money for nothing, change for free" approach [6], which we however deemed incompatible with the rather conservative client domains we are looking at.

Poppendieck and Poppendieck have presented a comprehensive classification of agile contracting schemes [7]. In their framework, adVANTAGE can be described as a multi-stage contract model with optional project scope. Larman and Vodde also discuss agile contracts in depth [8] – besides a strong focus on the legal details of agile contracts, they describe payment schemes that are similar to our approach, but more generic than the concrete model we present in the following section.

3 The adVANTAGE Contracting Model

To ensure fairness and efficiency for both parties in an agile development agreement, our contracting model combines elements of fixed-price and T&M contracting

models: adVANTAGE strives to provide some idea of the overall project scope (in terms of requirements, time and budget) to users and developers, as they would have in a traditional, fixed-price project, however without exposing the service provider to the risk of being committed to that exact effort. Instead, adVANTAGE ensures that the client pays exactly for what is delivered, as he would in a T&M situation – however without exposing him to the risk of a runaway project that never gets done, as the service provider will be equally encouraged to complete the project efficiently.

The key commercial principles of our contracting model thus are risk distribution and efficiency incentives – for both project parties, and for the whole project duration. To enforce them, adVANTAGE does not try to shoehorn a complete software project into one contractual framework, as traditional approaches do, but instead breaks contracting, pricing, inspection and payment down to the elements of the agile process. In the following subsections, we will show how these commercial aspects tie in with the sprints and deliverables of an agile process model:

Step 0: Initial requirements collection and budget estimate. To obtain an initial overview of the project scope and cost, we collect all the client's requirements before the first iteration. Typically, these are "must-have" as well as "nice-to-have" features, business goals as well as business ideas. Acknowledging that clients are typically unable to define their requirements in detail (let alone in formal specifications), we only collect them as "user stories", i.e. individual, testable features with a coarse, non-technical description on the business level.

The service provider then estimates the effort required for implementing each of the user stories. Due to the coarse nature of the user stories, these estimates are naturally subject to some level of uncertainty. However, we expect this uncertainty to be no higher than it also is in traditional bidding situations that providers have to deal with, and the uncertainty is distributed over a large number of individual user stories, instead of accumulated in one fixed-price bid. Still, the service provider needs sufficient domain knowledge to make competent estimates, and should perform this step in close cooperation with the client to avoid misunderstandings.

In contrast to traditional contracting models, the total of all user stories' effort estimates is not used to calculate a fixed price tag in the adVANTAGE approach, but serves as a plausible point of orientation for the following (iterated) steps:

Step 1: User story prioritization and sprint definition. Based on the service provider's price estimates and the clients' internal budget ceiling, the client can now prioritize, eliminate or add user stories to match his means and needs. In doing so, the client needs to balance the importance of each user story with his available budget and desired timeframe. The transparency of these trade-offs to the client is an important difference from traditional fixed-price models that often tempt clients to force service providers into aggressive project scopes and schedules because they won't affect the price anyway. Instead, the continuous budget focus encourages

clients to prioritize their desires by which user stories are most critical for putting the system into productive use, and which ones can be deferred. In addition, since this (re-)prioritization of user stories is possible after every sprint (as we will see below), the uncertainty in defining and detailing necessary system functionality is also distributed over the whole project duration (where its severity will gradually diminish), instead of being cast in stone at the beginning, when it is largest.

Based on the prioritized user stories, the client and service provider can now agree on the contents of the first sprint, i.e. the selection of the highest-prioritized user stories to be implemented next. To include users in the evaluation and prioritization of further sprints right from the beginning, we recommend that even the first sprint should include a sufficient selection of user stories that will yield a running – even if not complete – system.

Step 2: Sprint implementation. At the beginning of each sprint, its user stories are refined into more detailed specifications in close collaboration between the client and service provider. The implementation of these stories then progresses according to established agile development practices.

The duration of a sprint can be chosen flexibly based on the scope and nature of the project. However, given that our approach is aimed at large-scale projects, we recommend a sprint length of 4-6 weeks. Besides strict time-boxing of sprints, another rule is that the selection of user stories cannot be changed within a sprint, but only between sprints. This not only enables the developers to focus on a fixed set of requirements per sprint and arrive at a consistently integrated, thoroughly tested and running product after each iteration, but is also a prerequisite for the subsequent inspection and billing step.

Step 3: Sprint inspection and billing. In contrast to traditional contracting models, the adVANTAGE model ties the billing of services very closely to the agile process structure: At the end of each sprint, each user story is inspected individually for completion and acceptance (client sign-off), and the estimated and actual efforts of all accepted user stories are tallied. Depending on whether all user stories were implemented satisfactorily, and whether the sprint was completed within the estimated effort, this can lead to several different billing scenarios:

Underspend on sprint. As shown in the example in Table 1, the basis for the billing of each sprint is a lump sum for activities such as the requirements elaboration, the scrum master's work and other efforts that cannot be attributed to individual user stories (here, e.g., 22,000 EUR). Also on the bill is the initially estimated sprint effort, based on the service provider's usual daily rate (here, a total of 80 person days (PD) at 1,000 EUR each). We now contrast this to the sprint's actual implementation effort: If the team required less effort, only the actual effort is billed (in Table 1, for example, an underspend of four person-days leads to a 4,000 EUR reduction of charges).

Table 1. Underspend billing

Prio.	User Story	Completed & Accepted	Estimated Effort (PD)	Actual Effort
1	User Story A	yes	18	18
2	User Story B	yes	7	7
3	User Story C	yes	42	38
4	User Story D	yes	3	5
5	User Story E	yes	10	8
	Total Effort		**80**	**76**

Estimated price (80 x 1,000 EUR)	80,000 EUR
Underspend (4 x 1,000 EUR)	–4,000 EUR
Cross-cutting tasks (lump sum)	22,000 EUR
Sprint Bill	**98,000 EUR**

Overspend on sprint. On the other hand, if the effort was higher than expected, the additional effort is still billed, however at a considerably reduced rate that penalizes the service provider (in Table 2, for example, the five person-days of overspend are billed at only 600 EUR each). This way, the adVANTAGE model ensures a fair distribution of risks between both project parties: The client's risk is reduced since he is only billed for the actual effort, and the service provider's risk is reduced since he is still paid even if his initial estimate turned out to be too low.

Table 2. Overspend billing

Prio.	User Story	Completed & Accepted	Estimated Effort (PD)	Actual Effort
1	User Story A	yes	18	18
2	User Story B	yes	7	14
3	User Story C	yes	42	40
4	User Story D	yes	3	5
5	User Story E	yes	10	8
	Total Effort		**80**	**85**

Estimated price (80 x 1,000 EUR)	80,000 EUR
Overspend (5 x 600 EUR)	3,000 EUR
Cross-cutting tasks (lump sum)	22,000 EUR
Sprint Bill	**105,000 EUR**

One might wonder why we do not calculate the over- or underspend individually for each user story, but instead for the complete sprint. The reason is that this allows the service provider to spend effort saved on one user story on another story that needs more work, if necessary, without being penalized (as shown e.g. in user stories D and E). This is fair because what matters to the client at the end of a sprint is the overall state of the product and budget, not how the provider got to it.

Incomplete/unaccepted user stories. We still need to consider how to deal with user stories whose implementation was not completed in a sprint's time box, or whose delivered quality the client did not accept. In either case, the client has the choice of transferring these unfinished user stories to the next sprint, or to cancel them and thus eliminate them from the product.

As Table 3 shows, if a user story is transferred to the next sprint (e.g. user story B), neither its estimated nor its actual effort so far are included in this sprint's bill, but will instead show up on the next sprint's bill – then incurring an even higher actual effort (cumulated from both sprints), which will be penalized as described above. In case the client decides not to go ahead with an incomplete or unsatisfactorily completed user story and cancels it (example: user story E in Table 3), the work performed up to that point will still be billed as per the above rules.

This cancellation policy is reasonable as both parties know that work that was committed to also needs to be paid – and again, the risk is distributed as the service provider can rely on the client not cancelling already completed work on a whim, while the client has the option of cancelling work on user stories when he feels that their maturity is sufficient or further development on them is not economical.

Table 3. Partial billing

Prio.	User Story	Completed & Accepted	Estimated Effort (PD)	Actual Effort
1	User Story A	yes	18	18
2	User Story B	no, transfer	~~7~~	~~14~~
3	User Story C	yes	42	40
4	User Story D	yes	3	5
5	User Story E	no, cancel	10	8
	Total Effort		**73**	**71**

	Estimated price (73 x 1,000 EUR)	73,000 EUR
	Underspend (2 x 1,000 EUR)	–2,000 EUR
	Cross-cutting tasks (lump sum)	22,000 EUR
	Sprint Bill	**93,000 EUR**

Iteration or termination. After each sprint, the client then has the option to start the next iteration from Step 1 again, where he can re-prioritize all user stories, and also eliminate or add new stories. Change requests for already completed and accepted features will be treated as new user stories. Since this re-prioritization of the user stories will again be guided by the question of what is most critical for the system to enter productive use, and how this can be traded off with the available budget and time, the project will be focused on its overall goals and constraints in every iteration.

Alternatively, the client can terminate the project after any sprint, when he feels it has reached the required functionality and/or its budget limits. Since every sprint results in a running system, this exit strategy is again risk-free for the client.

4 Industry Example and Conclusion

The adVANTAGE model has been adopted in practice by adesso AG, a large German software development company mainly serving the public, financial and healthcare sector. In an ongoing large-scale project, adesso is building a new contract management system for a German life insurance company.

Given the complexity of the system – both in terms of its business requirements and its integration requirements with the legacy infrastructure – both partners agreed that a reliable up-front fixed price estimate would be very difficult. By employing the adVANTAGE model, the partners could however break the project down to the insurer's individual user stories, and proceed with incremental implementation and billing of more manageable chunks that could be prioritized according to the insurer's wishes for going live. With the project still ongoing, a visible benefit of the approach so far has been that the collaboration between adesso and its client is more focused on progress with the individual user stories, than on questions regarding components throughout the system (which could otherwise distract attention from the essentials as developers are drawn to problems that are interesting but not immediately relevant). As a formal basis for billing each sprint, staff members keep timesheets just as they would for traditional projects, so the administrative overhead is low. The actual efforts up to now also agree well with the initial estimates, indicating that adVANTAGE gives no incentive to either party to blow up efforts.

In discussing the above example and our approach in general, it should be noted that an understanding of the risks of agile software development, and a degree of mutual trust in the business partner's competence, quality standards and business ethics, and a commitment to open communication is still required for a flexible contracting model such as this to work. Given these prerequisites, however, we believe that the risk-sharing and fair billing provisions of adVANTAGE will help agile processes to be more readily adopted by third-party software suppliers and their clients alike.

References

1. Curtis, B., Krasner, H., Iscoe, N.: A Field Study of the Software Design Process for Large Systems. Comm. ACM 31(11), 1268–1287 (1988)
2. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall (2002)
3. Boehm, B., Huang, L.G.: Value-based Software Engineering: A Case Study. IEEE Computer 36(3), 33–41 (2003)
4. Lehman, M.M.: Uncertainty in Computer Application and its Control through the Engineering of Software. J. Softw. Maint: Res. Pract. 1(1), 3–27 (1989)
5. Van Cauwenberghe, P.: Succeeding with Agile Fixed Price Contracts. In: Aguanno, K. (ed.) Managing Agile Projects. Multi-Media Publications Inc. (2005)
6. Sutherland, J.: Money for Nothing and Your Change for Free: Agile Contracts. Agile 2008 talk, `http://www.slideshare.net/gerrykirk/money-for-nothing-agile-2008-presentation`
7. Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit. Add. Wesley (2003)
8. Larman, C., Vodde, B.: Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum. Pearson Education (2010)