

Pay Per Use Contracts

Nora Sleumer, Massimo Arnoldi, Massimo Milan
Lifeware SA, Switzerland
www.lifeware.ch

In a Pay Per Use Contract the software supplier is paid in proportion to how often the software is used, so the customers and suppliers share the risks and the long-term profits. With such a contract, customers can ask for new features without having to pay everything ahead of time, allowing them to move faster and be more competitive. A team of programmers earns according to the revenue-generation value of the code instead of only once for writing the code. The interests of the customer and the supplier are thus aligned so they can work together as partners. This is a perfect framework in which to apply XP since the contract is simple, allows room for flexibility, and provides an optimal basis for the planning game.

1. Introduction

Imagine the following scenario between a software supplier and their customer who apply XP:

Customer calls: "We're getting a new sales line. But they want special feature X. By when do you think you'll be able to get it done?"

Supplier: "Hmm, how many units do you reckon you'll be able to sell?"

Customer: "We figure about 200,000 a year, with 10% of the profits for you as usual."

Supplier: "Well, that sounds worth it! We better get going on it right away. That other feature A we had planned for this month can wait until afterwards, right? And maybe we can borrow the X-Guru from the other team for a week. In that case we should be able to have it done by the end of the month."

Customer: "I agree that this is more important than feature A. End of the month is perfect. I'll fax you the updated contract including this new sales line right away."

Why did the supplier so readily agree to rearrange the priorities and wage organizational battles over who works on what? Because she knew her company would probably get a good percentage of the 200,000 a year, making it worth it. Her company doesn't get paid according to how difficult it is to implement a feature, but according to how much revenue the feature generates.

One year later:

Supplier to her boss: "Here are the reports of the new sales lines."

Boss: "Our customer didn't quite achieve the sales goals for feature X, did they?"

Supplier: "No, but the profits of Y and Z more than make up for it."

Pay Per Use Contracts allow customers to take more risks because they don't have to pay for the development up front. The more products they develop together the more the risk is spread. Suppliers still profit from windfall profits if the feature generates revenue out of proportion to the difficulty of implementing it.

2. Pay Per Use compared to other Contracts

Classical contracts, or *Fixed Scope Contracts*, as they are called in the paper "Optional Scope Contracts"¹, provide incentives to the supplier to do the job as quickly as possible with the least effort while just barely meeting the customer's minimum requirements. The supplier tries to sell new features for the highest price possible. What the supplier gains, the customer loses. *Optional Scope Contracts*, where the time, cost and quality are fixed but the scope is variable, motivate suppliers to meet customer's quality requirements, but not necessarily to do more than the minimum required. The customer still needs to decide whether the return on investment is good enough. (Was that perfect login page worth it?) In these two scenarios the avarice or laziness of the programmer is balanced by the need to convince the customer to pay another round.

Pay Per Use Contracts, on the other hand, better align customer and supplier interests because they share the profits and the risks. In the planning game the customer and the supplier work together to determine the time in which a new feature can be implemented given the available manpower and difficulty of the implementation. The supplier won't try to inflate the difficulty of stories because they don't factor in the contract. Suppliers don't need to maximize short-term profits if they know they will share in the long-term profits, assuming that they can finance the development costs in advance. Customers know they can depend on the suppliers to do what's best for both of them. Similarly to Pair Programming, working together as partners reduces the failure rate of a project.

¹ Kent Beck and Dave Cleal, "Optional Scope Contracts", 1999

The idea of Pay Per Use is not new. In “The Age of Access: The New Culture of Hypercapitalism”² Jeremy Rifkin imagines a society where access to nearly every aspect of human life is accounted for. Brad Cox applies the Pay Per Use model to the market of pieces of software, in “Superdistribution: Objects As Property on the Electronic Frontier”³. Others are less convinced of this model where micropayments are concerned⁴ but Apple, for example, has just opened its online music store⁵ where songs can be bought for 99 cents. At a lower level of granularity, in our experience, Pay Per Use matches the XP process very well.

3. Contract Structure

The simplest version of a Pay Per Use Contract is one where each time a product is used, the supplier gets paid a fixed amount. This can be divided into several components. For example, a company providing third-party life insurance management could be paid:

- each time a new policy is taken out
- a percentage of the total value of the policy
- a part of the fee the customer has to pay to modify the policy
- a yearly fee on a per-active-policy basis

However, Pay Per Use contracts only work for projects where the overhead for determining how often features are used is not out of proportion to the cost of the software itself. They also make more sense for projects where a long-term collaboration between both parties is given than for a one-off piece of software requiring no maintenance.

Given the four parameters Time, Cost, Scope, and Quality – which ones are fixed in the contract? Instead of leaving only one variable, all four could be under continual discussion within a broad framework. If the customer absolutely needs a certain feature by a certain date then the supplier can increase the manpower. If the customer wants too many features too quickly then the quality will probably be sacrificed, and the customer will feel it later. If the scope of a feature is larger than originally estimated, then the time can be moved, or a basic version could be implemented initially, with additional functionality to be added later.

Depending on the type of product some of the parameters may be more important than others and therefore require a more detailed description in the contract. Because the interests of both parties are aligned to a large extent, the risks of litigation are lower.

Since the contract is a long-term contract a support contract could be included in the basic contract. The supplier has to be aware, however, of how many new development requests could be hidden in the support requests. It is a matter of negotiation what constitutes a fair share of the profits compared to the implementation and support work done by the supplier.

Trust between both parties can be built up by working with a series of contracts. The first one is of course the most risky for both sides. In the following contracts the customer and the supplier will adapt the new requirements according to the collaboration experience.

4. Communication between the customer and the supplier

Focusing on the long-term revenue-generation aspect gives the whole team a strong basis for discussing the importance of features. The client and the supplier share the same vocabulary because they are both end-client driven and not technology driven. By measuring the costs and probable revenue you can compute profitability and use it as a basis for determining the priority list.

Since the suppliers are in it for the long run, a virtuous communication circle is created that keeps the software alive. A positive feedback loop of user suggestions and the long-term use stimulate the implementation of the important features at a high level of quality. On the other hand, since quality isn't the most important aspect, the quality / new functionality trade-off can be kept in mind.

5. Cost Analysis

For the customer:

The cost and effort of initiating a new project are reduced if the customer doesn't have to pay for the whole development up front and is not inhibited from innovating along the way. The customer doesn't have to pay each

² Jeremy Rifkin, “The Age of Access: The New Culture of Hypercapitalism, Where all of Life is a Paid-For Experience”, J. P. Tarcher, 2001

³ Brad Cox, “Superdistribution: Objects As Property on the Electronic Frontier”, Addison-Wesley Pub Co., 1996

⁴ Clay Shirky, “The Case Against Micropayments”, The O'Reilly Network, 2000

⁵ <http://www.apple.com/music/store/>

time they change their mind and so are not forced to stick to a bad plan. The clients don't have to pay through the nose for each post-project modification or support contract because they are inherent in the contract.

For the supplier:

The suppliers are assured of a constant income flow over many years as opposed to project-based payments.

The supplier's administration costs are lower because they don't have to provide proof of billable hours.

On the one hand you don't want to inhibit the customer from innovating in order to get a bigger market share with the product. However, the customers could be nitpicky and demand so many changes that the effort to create the final product is no longer proportional to the revenue it generates.

6. Risk Analysis

The risks associated with the difficulty of estimating the costs for developing software are carried by the supplier (they may need to add more manpower), the customer (they may have to reduce scope or move the date), or both (they will both suffer from the absence of quality). If the customers change their mind to such an extent that the development costs are not likely to be recovered by the long-term profits then the supplier has a problem. In a series of contracts the supplier could then demand a higher share of the profits for the next contract. The supplier could also just refuse to implement the changes.

Determining what constitutes a fair share of the long-term profits compared to a fair share of the risk requires experience and skill.

For the customer:

The customers have a better chance of getting the software that they really want instead of that which they thought they wanted because the supplier is also interested in providing a maximal revenue-generating product. The specifications don't have to be written in stone in every detail – the details can be adapted along the way since the supplier isn't interested in minimizing their short-term costs as much as maximizing the long-term profits.

The chances that the supplier goes out of business and the customer is stuck with a dead piece of software are lower because the supplier wants to be around to share the profits forever.

For the supplier:

What if the software supplier cannot judge the risks involved, such as the first time that such a contract is discussed? In this case a guaranteed minimum could be agreed upon. This mitigates the risks for the software supplier. The customers could even go bankrupt before creating enough revenue to cover the supplier's development costs. If this risk is not unlikely, then the minimal guaranteed amounts could be paid in several installments.

If the guaranteed minimums for several features are combined then the risk is spread, profiting both parties. The guarantees can also be compensated from year to year. As long as the average success rate is high, the supplier will likely be willing to share the risks.

On the other hand, the higher the minimum guarantee, the more it looks like a standard contract and the less the supplier is interested in optimal resource allocation. The customer will probably want to lower the share of the long-term profits too. Both sides need to consider the risk/profit trade-off.

7. Acknowledgements

We would like to thank Kent Beck and Lauro Canonica for their insightful comments.

8. Conclusions

A Pay Per Use Contract seems to be the ideal mechanism to encapsulate XP. Since the interests of the customers and the suppliers are aligned, the complexity of a new feature can be discussed honestly, removing the inhibition of adapting the system as it is developed. Customers can take risks more easily and can be sure that the supplier handles in their interest. The suppliers get a share of the long-term profits. Even though the customer and the supplier look at the situation from two different points of view, they look in the same direction.