IT Partnership as a Service. A new model for IT collaboration.

Abstract

The contemporary models of billability by the hour and through a fixed price are intuitive models that follow through basic logic and in that sense find their strength in the simplicity of its financing model. The models do not account for – or incorporate – important aspects, however, such as motivation, incentives and trust. In real life all of these aspects are present in software development projects and all other business projects.

The main critique of the contemporary model arise from these aspects and its disadvantages lead to high costs for the client and dissatisfaction concerning the adherence of requirements due to the lack of incentives for software vendors to produce high quality software.

We propose a new business model *IT Partnership as a Service* (ITPaaS) based on the core values *commitment*, *involvement* and *transparency* which cover the many relational and social aspects that are needed in business cooperations. The other strength of the *ITPaaS* model is its adherence to requirements, to produce high quality results and go further than what is required because of its incentive model: vendors become stakeholders and charge no costs for development. This makes it possible for vendors to become product owners and for clients to start without (large) initial investments.

VipaHelda, Rotterdam, the Netherlands, February 24th, 2016

This document can be downloaded at http://www.vipahelda.com/itpaas

Introduction

Software development has been in the market for decades and now more than ever it is booming [3]. Times have changed and so has the infrastructural landscape: starting with computers the size of a room, developments swiftly progressed from web to mobile development and now even to the Internet of all things, ranging from home automation with wireless light bulbs to health-care technology in our clothing and accessories and such. The business model of software development, however, has not changed.

Current software development models are heavily reliant on (hourly) billability, which varies greatly in actual business cases. This not only leads to a great deal of inefficiency and administrative overhead, but also to much frustration, and negative impacts on trust and personal relations.

In this paper we will examine real-life software development models and propose a radical change in contemporary IT business models. One that will ultimately prove to be more robust and sustainable, more cost-efficient and effective, with a potential bigger market penetration and increasing the quality of business relations.

We will show that software ownership does not have to cost much and can compete with concurrent business models and even exceed these in many fields, given the right conditions.

In the next subparagraph we dive a little bit deeper into the conventional model of hourly billability.

Experiences

Billables by the hour in general has a bad aftertaste. It is never pleasant to be surprised by extra costs during settlement of the account and especially not for seemingly irrelevant and small items such as meetings, phone calls, e-mails or (short) troubleshootings.

This conventional system of hourly billables is extremely inefficient [1]. Bug fixing and enhancements – big or small – all need to be approved. Even worse, decisions and priorities are often made based on costs, which at first instance seems like a wise thing to, but can ultimately lead to overwhelmingly negative effects when the long term effects on the application and business processes has not been taken into account.

Our own experience provides many situations where we came across these kinds of examples with disastrous results on company-wide productivity. We provide an illustration where small decisions have a cascading effect. An unnamed company decided to disapprove the enhancement of a specific function of an app for cost-related reasons. Though the lack of fine-tuning of this function only slightly irritated the user in the beginning, the effect was that the task that this specific function was supposed to do, was done manually instead. In effect, all related tasks had to be done manually and the contributing value of this app was as good as negligible.

Unfortunately, this illustration is only one of numerous real-life examples. Where this decision led to a dysfunctional app in the sense that the app was not used, other examples exist where cost-based decision-making led to much time wasted in the development process, where the software application was not suited for actual implementation and the presence of frustrations from the beginning of development until after delivery of the application. Conventional billability models have not proven that they can stand up to the needs businesses and consumers have.

In the next paragraph we examine the strengths and weaknesses of contemporary business models and software methodologies. We will also propose a new business model. In Paragraph 3 we will show the results of the practical implementation of model and in Paragraph 4 we give our conclusions and suggestions for further research.

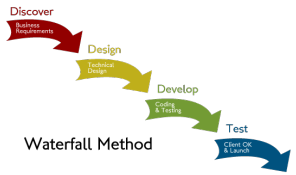

Figure 1: The waterfall method is straightforward and intuitive and works well, given the resources are available and initial requirements are foolproof.

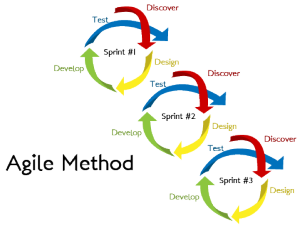

Figure 2: The agile method is an extensive yet simple model that has proven its effectivity and its capability to deal with errors in the process, but requires much resources to make it work well.

Methods

In the previous paragraph we discussed the general problem of hourly billability in software development. In this paragraph we will dive deeper on the strengths and weaknesses of contemporary models.

We evaluate models by the needed input and delivered output. In other words, what are the costs in terms of money and time, and what are the results in terms of adherence to requirements and on-time delivery?

Model evaluation

Even though it is hard to fully separate the financial aspect from the requirements needed and the quality of the implementation thereof, we will make a short overview of the finance and requirements.

Finance

As an alternative to the hourly billability model many projects make use of the fixed price model. Problems concerning billability would partly be solved in this way, but still there are a number of practical issues that are not covered. The issues that come to mind are as follows:

1. If the total cost of the project exceeds the fixed price, the software company has a loss. In the other case, the client is actually paying too much.
2. In the first scenario of a fixed price arrangement where the software company has a loss, if the client decides to want some functional enhancements – once again – for a fixed price, the software company would be inclined to incorporate the losses from the first project into the fixed price of the second.

Regardless of the model, we should note the importance of the underlying software development methodology and its impact on the financial aspect.

Software development methodologies: Waterfall and SCRUM. Contemporary models all work to a certain extent. The key difference between success and failure of the practical use of these models is not so much found in the models or methodology itself, but rather in the contextual implementation, i.e. the context of the company, its size, and the customer it is dealing with. In this specific case we believe size really does matter.

Both Waterfall and Agile/SCRUM are software development methodologies incorporated in a given business and earnings model [2].
Whereas the Waterfall model (see Figure 1) is successful when requirements are clearly written and understood by every cascading party – which in practice is hardly ever the case – in practice the Agile/SCRUM model as in Figure 2 works much better because of its short sprints and close interaction between the developer and the client that make timely intervention possible. Close and frequent communication, heavy interaction and a multidisciplinary team are key features and strong points of a SCRUM implementation. The same factors, though, are what causes projects implemented in this way to be very costly.

For this reason successful SCRUM and Waterfall implementations are almost always seen in large companies with big budgets for software development projects, which have much financial flexibility.
The question arises then: *How do smaller businesses cope with software development?*

Requirements

An evaluation of the adherence to requirements is probably best shown through the following practical and familiar example.
A banking company hired a software company to build an app for its users. After a year, the app was still not fin-

ished. Many issues were visible in both the architecture of the software as well as in its functionality. It turned out that the app was not what the banking company had in mind and the following year they would start over and build a new app. In a sense this was good news for the software company as they would have another opportunity to make money. For the banking company on the other hand it was frustrating. They even considered charging the software company for this failed app and letting another party build the new app.

Analysis of the contemporary model

The example above shows that that it is wise to evaluate the contemporary model before rolling into the process of software development, in order to know what one may expect.

We analyzed the contemporary model in a number of real life situations and found three factors that caused the difference between fail or success of the project. Each of these three factors is crucial and necessary to co-exist with the other. Not coincidentally in all of our analyzed business cases at least one or more factors were either neglected or not present.
The three factors that we discovered in our findings are the core values of business: *commitment*, *involvement* and *transparency*. We provide a short overview of how these core value play a role in the success of a business model.

Core values of business models
Commitment. Almost every software development company has a limited liability, that is, as soon as the developed software application is delivered and approved, all responsibilities for the software company end. This leads us to a second point.
Involvement. The software company within the contemporary business model has no incentive nor benefit in whatever happens with their developed application and how it is used for the business. There is no impact on the software company whether the scope of the requirements for an application is fine-tuned according to the clients needs. The project scope is a given: what the client wants, the software company gives. Conversely, the software company could be inclined to stimulate the client for more (complex) requirements and more functionalities than needed, as that would even benefit them [4]. On the long run, however, it would not benefit personal relations.
Transparency. The previous core values *commitment* and *involvement* are related with this core value as well. For a company to commit to an agreement with a significant amount of time and resources and to allow involvement from another party in its own business processes, requires a lot of trust. Trust is often limited by the amount of transparency, and so transparency becomes the hinge on which commitment and involvement turn.

The existence of billability by hours and clauses for every specification or enhancement creates a formal way of working yet, not a partnership that involves trust and commitment. Billability leads to defining clauses; clauses lead to discussions about what is inside or outside of this definition and discussions – in this sense – are not beneficial towards a long-term partnership.

A new model: IT Partnership as a Service

Given our findings on the three main ingredients for a successful software development projects we propose a new business model for – but not limited to – software development.
The model is called *IT Partnership as a Service* (*ITPaaS*). Within *ITPaaS* the relationship between client and vendor becomes like that of partners, that is both parties are stake-

holders of this business. Both parties are actively involved in the conception of the product and they are committed to help each other and the business to their best.

In its purest form *ITPaaS* consists of the following steps.
Conception. The client wants a software application and the vendor is willing to build this. The vendor wants to take it a step further and wants take a stake in the project.
Administration. The vendor has full transparency on the (financial) data and plans of the customer. The client has full transparency on the resources and capabilities the vendor has available for the project. This helps both parties to commit themselves to the partnership.
Ownership. **Both** partners have ownership of the product. The vendor is involved in the product as well and gets a say in the conceptualization of the product.
Financing. If the vendor has faith in the product, he offers to build the app for **free**. Both partners then agree on an earnings model such as earnings based on the number of users.

We emphasize that the vendor can only agree to enter the partnership when he has a good feeling about the prospects of the product, seeing the fact that he makes the initial investment of **zero development costs**. The transparency factor partly helps in that, but the involvement in the product conceptualization strengthens his belief in the success of the product.
Also note that the incentive for the vendor lies in the success of the product, which is a key difference with conventional models!
The three core values recur in this model where *transparency* and *involvement* are visible in the steps *administration* and *ownership*, respectively, and *commitment* is shown through the chosen earnings model. Not only do the core values occur but they are necessary conditions in order to make *ITPaaS* succeed.

In the next paragraph we describe a best practice example of *ITPaaS* that incorporated these three core values.

Results

In the previous paragraph we showed why most business models do not succeed in creating efficient solutions for software development, because they are missing one or more of the three core values commitment, involvement and transparency. We also introduced the *ITPaaS* model and showed that this model does contain these core values.

In this paragraph we show the results for a furniture company that decided to use *ITPaaS* to create a software application that would not only make their business processes more efficient, but would also change the whole process workflow.

In this real-life example there is a small furniture company that sells configurable cabinets, which can consist of a number of materials and colors, each of them having a number of accessories, such as grips and boards.
The whole workflow from customer visit to sales consisted of multiple actions such as the creation of designs for the customer, quotes, bill-of-materials for the actual order, and lots of e-mailing in between and much time before a single cabinet was installed at the customer.
This furniture company approached a software company for a full-fledged software system, including a back-end in the cloud and applications for mobile devices, which had to be able to process all administration of orders, designs, but also show 3D views of every cabinet and combinations of cabinets. Most of all, the software had to be generic as the furniture collection could easily be expanded – new collections of cabinets would easily have to be added, even to

other categories within furniture.

The software company performed a thorough calculation and came back with a prospected price of €100,000. Though the furniture company was doing good business and growing, it did not have the budget for such a large investment. The software company understood their problem and thought of the following solution.
The two companies had done business before for a 2D application meant for viewing and configuration and there was mutual trust. Both companies were also aware of each others capabilities and future prospects and committed to a long-term relationship where the software company would be co-owner of the products well. In short, the necessary conditions for *ITPaaS* were met.
The software company had to do a risk analysis before it could set the final step. It analyzed the existing software components and libraries and how this would be implemented in the new application. Because they previous project for the furniture company was set up with a robust software architecture there were many reusable components. These arguments as well as an estimation of the return on investments led them to make the decision to invest in the partnership.
Both parties – *partners* now – came to an agreement concerning the earnings models. The software company saw much potential in the software application and decided to go for a SaaS solution. There would be no development costs, but a licence model would be used: a mere €50 for each user per month. The software company could then white-label the product as a generic 3D configuration app and was able in this way to both provide a solution for its partner as well as find a way to increase its own revenues.
The resulting app was called the **BTO app** about which more information can be found at `www.btoapp.com`.

The actual implementation of creating the application did not go without troubles. Because both partners were product owners now, both had a say in the requirements and seemingly small enhancements that the client wanted, had to be explained by the vendor that technically seen were large operations.
There were also different views on the requirements necessary to create the minimal viable product and many discussions, but the commitment to make this work and continuous transparency supported the trust both parties had in each other to make this application successful.

Conclusion

In the previous paragraphs we discussed and analyzed contemporary business models for software development and proposed a new model *ITPaaS*: *IT Partnership as a Service*.

The contemporary model of billability by the hour or through a fixed price is an intuitive model that follows through basic logic and in that sense finds its strength in the simplicity of its financing model. The model does not account for or incorporate important aspects, however, such as motivation, incentives and trust. In real life all of these aspects are present in software development projects and all other business projects.
The main disadvantages of the contemporary model arise from these aspects.

From a financial perspective contemporary models leave much room for improvement as well. Costs can rise high and there is no incentive for the software company to keep costs low.

The *ITPaaS* model is based on its core values *commitment*, *involvement* and *transparency* which cover the many relational and social aspects that are needed in business co-

operations. The other strength of the *ITPaaS* model is its adherence to requirements, to produce high quality and go further than what is required because of its incentive model: vendors become stakeholders and charge no costs for development. This makes it possible for vendors to become product owners and for clients to start without (large) initial investments.

ITPaaS in itself is separated from the software methodology used, such as Waterfall or Agile/SCRUM, and not limited by the costs implied by the methodology because of its particular financing model. Given the important aspect of financing in *all* business models, whether conventional or radical, we believe that *ITPaaS* is especially suited for small and medium sized companies, having no large initial investment capital.

The *ITPaaS* model does, however, require the three important and necessary conditions defined by its core values and confidence in a successful launch of the product.

Further research
Besides a licence model other models are possible as well. It is out of the scope of this white paper to discuss or research all of them, but we will mention a few possibilities here.

- Pay per use (API usage, bandwidth, etc.)
- Pay per transaction (mobile payment, etc.)
- Pay per reservation
- Pay per user
- Pay per unit (product, device, etc.)

All of these have its pros and cons depending on the specific product/application. In our experience the main blocker is the lack of information. Even if an *ITPaaS* Partner is willing to give insight in all of his "books" or Google Analytics, there are some things that are simply not registered.

A solution could be to estimate these by extrapolation or other mathematical methods. This too is out of the scope and requires further research.

References

[1] The truth about the billable hour. https://www.law.yale.edu/student-life/career-development/students/career-guides-advice/truth-about-billable-hour.

[2] Choosing the right methodology. http://gera-it.com/approach, 2016.

[3] R. Miller. Digital transformation requires total organizational commitment. http://techcrunch.com/2016/01/31/digital-transformation-requires-total-organizational-commitment, 2016.

[4] R. D. Rotunda. The problem of inflating billable hours. https://verdict.justia.com/2014/11/17/problem-inflating-billable-hours.