Agility in Fixed-Price Projects

By Siju P. Varghese

Executive Summary
Corporate IT lawyer Alistair Maugham is one among many who argues that agile development is ill suited to government IT, and more generally for fixed-price projects. A fixed-price project typically locks down the scope, cost, and time of the project and leads us to believe there is no room for agility. However, if a proper delivery structure is established between the customer and the provider, and both parties believe in the Agile Manifesto, the customer may gain the best of both worlds — security and control of the fixed-price project and the ability to respond to changes in the market.

Commonly Used IT Development Contract Categories
Following are some of the contracts engaged in IT development projects:

- Fixed-price contracts: This type of contract involves a fixed total price for a well-defined product. A fixed-price contract is a contract in which the amount of payment does not depend on the amount of resources or time expended. Such a scheme is often used by military and government contractors to put the risk on the side of the vendor and control costs. Scope, features, planning, and timing are fixed in this type of contract.
- Cost-reimbursable contracts: This category of product involves payment to the seller for the seller’s actual cost, plus a fee typically representing seller profit. Cost plus fee (CPF), cost plus fixed fee (CPFF), and cost plus incentive fee (CPIF) are variations of the cost-reimbursable contract.
- Time and material contracts: These contracts are a hybrid of fixed-price and cost-reimbursable contracts. This contract can grow in contract value like cost-reimbursable contracts; conversely, it resembles fixed-price contracts in that the buyer and seller fix the rates for resources upfront.

Why Do Customers Often Prefer Fixed-Price Contracts?

- If projects are awarded after a multi-provider bidding process, the customer knows the scope, timing, and price required to choose among the bids.
- Customers think they take no functional risk: if the provider does not deliver on time or according to specifications, the customer can often take legal action. Of course, if the customer really needs the software on the given date, he or she is also in trouble.
- Customers think they take no financial risk because the price is known beforehand; however, surprisingly, many “fixed-price” projects cost more than initially agreed upon.
What are the risks involved in fixed-price contracts?
The obvious risk is on the side of the provider. If the estimates are wrong, the project will lose money. Less obvious risks are the change request game through which the provider negotiates additional revenue through scope changes. If the supplier had poorly underestimated the effort or risk or quoted an unrealistically low price, the losses could even threaten the existence of the supplier, which also presents a problem to the customer.

Introducing Agility into Fixed-Price Projects
Under a fixed-price project, the customer has the security of fixed scope cost and schedule; in addition, what if the customer also gets the following benefits:
- Ability to get the value of the system sooner by implementing crucial features upfront
- Provision to delay the decisions on certain features
- Provision to replace the irrelevant features with more relevant features even after the project is started
- See the real progress of the project earlier in the phase and provide feedback

This is the best of both the worlds; however, let us first examine how we can provide the above-mentioned benefits to the customer within the boundaries of the fixed-price contract, by highlighting the Agile Manifesto:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Let’s examine two key phases involved while engaging in a fixed-price contract, and how agility can complement the contract to provide more value to the client: **Selling** and **Implementing**.

### Activities in the Selling Phase of Fixed-Price Projects
When the provider decides to bid for this fixed-price project, this is the time to establish a fixed scope, schedule, and cost. An RFP sometimes does not provide much clarity on the intended features of the project. Developing a scope statement is the first part of the game, which should contain project objectives, product scope description, project requirement,
project boundaries, project deliverables, product acceptance criteria, project constraints, and project assumptions.

Taking the project scope statement as input, the provider performs various duration estimation tools and techniques, such as analogue estimating, parametric estimating, three-point estimating, and so forth, to estimate the complete project duration. The provider will add a fixed percentage slack to handle some of the known unknowns. Similar techniques are used to estimate the cost of the project.

At the end of this phase, a proposal will be created to communicate that: *We will deliver* \( n \) *number of features (those that were requested in the RFP), on so-and-so date, for* \( x \) *amount of money.* In addition, a complete project plan will be provided with commitments to interim deliverables.

Let us focus on the potential delivery structure for a 24-month, fixed-price traditional project with a typical waterfall software development methodology.

In the delivery model in Figure 3, the customer is seeing the implementation of the most crucial feature during the 21st month of the 24-month project, and the end user has to wait until 24 months before he or she can take advantage of the modified system. Until then, the progress of the project is measured based on the completion of the activities and not based on the completion of the features.

**Modified Selling Phase to Bring Agility**

One important tip we need to remember when selling a project is that multiple small projects are better than one big project. Minimized scope is one of the top five recipes for success, according to *Recipe for Success: CHAOS Ten:*

*Time is the enemy of all projects, and since scope affects time, or project duration, they are linked. Clearly then, minimizing scope increases a project’s chances of success. Minimized scope has replaced small milestones. While these two factors are similar, the act of minimizing scope leads to greater success than does creating smaller milestones.*

What if the customer is looking for a large-scale project that will span across multiple years? First of all, we have to prioritize the requirements: what is crucial, what is important, and what is nice to have? If we just do the crucial stuff, could the customer use the product? If not, what do we need to add? What would be enough for a first, useful release?

Given that the customer has the secure feeling generated from the fixed-price contract, why would the customer need to prioritize the features? Some of the advantages are listed below:

- The users get the software with crucial features earlier than expected.
- Project risk is reduced as we work on fewer requirements and concentrate on the high-value features.
- The customer can evaluate the outcome of the project sooner.
- The customer can delay their decision about the requirements that are not in the first release. At that time, they will have more information and knowledge to make a better decision. This allows the customer to “decide later.”
- Project cost is reduced if the customer needs to drop or postpone some features.

Breaking down the scope into crucial, important, and nice to have requirements enabled us to have a slightly altered delivery structure, as shown in Figure 4.

The competitive advantages of this deliverable structure are obvious: the customer can start using the system earlier and the crucial and most important features are implemented earlier in the life cycle. Moreover, the progress is measured based on the working software as compared with comprehensive documentation (Agile Manifesto).

---

**Figure 3. Waterfall delivery model of a fixed-price project.**

The customer did not see working software until the 21st month.
Figure 4. Potential delivery model for a fixed-price project with agile techniques. Production-ready software with highest priority features is integrated at the end of the iteration.

In this delivery model, the cost incurred in the “nice to have” and less important features at the time of initial release is minimal to none. However, in the traditional delivery model, we would have incurred a cost to plan and design software for even a “nice to have” feature, because this was a fixed-price project. Why is this important to my customer? Let us proceed to the implementing phase to answer this question.

Activities in the Implementation Phase of Fixed-price Projects

At this stage, the goal of the project manager is to deliver mutually agreed upon features (scope), at an agreed upon date, and within the budget — nothing more (according to A Guide to the Project Management Body of Knowledge (PMBOK® Guide), anything more is gold plaing) and nothing less. If everything goes as per the original plan and there are no changes in scope and schedule, the customer got what he or she asked for. The Standish Group CHAOS Report for 2009 shows only a 32% success rate in IT projects (see Table 1).

Why were 68% of projects either challenged or failed in 2009? Although we do not have detailed data on why these projects were challenged, following are some potential reasons:

- Inaccurate estimation due to incomplete RFP or lack of expertise
- Underbidding to beat the competition
- Invalidated scope definition due to the impact of various environmental factors — some of the originally defined scopes may not be important anymore, but maybe new features are required
- Impacts of change request on schedule — As change control board approves more and more change requests (due to the right reasons), the implementation of the project will continue to be delayed

<table>
<thead>
<tr>
<th></th>
<th>Year 2009</th>
<th>Year 2006</th>
<th>Year 2004</th>
<th>Year 2002</th>
<th>Year 2000</th>
<th>Year 1998</th>
<th>Year 1996</th>
<th>Year 1994</th>
</tr>
</thead>
<tbody>
<tr>
<td>Successful</td>
<td>32%</td>
<td>35%</td>
<td>29%</td>
<td>34%</td>
<td>28%</td>
<td>26%</td>
<td>27%</td>
<td>16%</td>
</tr>
<tr>
<td>Challenged</td>
<td>44%</td>
<td>19%</td>
<td>53%</td>
<td>15%</td>
<td>23%</td>
<td>28%</td>
<td>40%</td>
<td>31%</td>
</tr>
<tr>
<td>Failed</td>
<td>24%</td>
<td>46%</td>
<td>18%</td>
<td>51%</td>
<td>49%</td>
<td>46%</td>
<td>33%</td>
<td>53%</td>
</tr>
</tbody>
</table>

Table 1. The Standish Group CHAOS Report on 2009 project success rate.
• Limited or no visibility of any of the implementations of even crucial features, until the last stage of the project, and if there’s any difference in expectation, it is too late at that point
• Potential relationship meltdown during the dispute of “It’s in the spec! No, it’s not! Yes, it is…” (i.e., disagreements about was agreed on but not implemented)

Does agile provide a silver bullet to solve all these problems and guarantee that the project will be successful 100% of the time? Absolutely not! However, let us examine how we can implement some of the agile techniques in a fixed-price project to alleviate some of the project risks.

Agility in the Implementation Phase of Fixed-Price Projects
As we execute modified activities during the selling phase, we enter into the implementing phase with a prioritized (crucial, important, nice to have) feature list; in Scrum terminology, this is nothing more than a product backlog. However, for large-sized projects, we may further sequence the features; ideally, our customer is the best choice for doing this exercise. The question is: What is the most important feature with complete details available among the crucial category? Can our customer force rank them? Here comes the second Agile Manifesto: customer collaboration over contract negotiation.

Are we omitting a big piece of the puzzle — change requests? Change requests are inevitable in many projects, especially in longer duration projects. The change requests could arise for various reasons, such as:
• Changes in environmental factors
• Changes realized after visualizing and/or experiencing the initial version of the software
• Changes due to the originally incomplete requirements or specification
• Changes because the originally established priority is invalid due to various external factors

In traditional project management, the PMBOK® Guide outlines the process to handle change requests as part of the Project Integration Management Knowledge Area. As the change control board approves the change, the scope is added to the project; in effect, the fixed-price contract suddenly becomes not so fixed. As one of the triple constraints (scope) changes, which will impact the schedule, so the 24-month project becomes a 26-month project, and the end user continues to wait for the new system for another two months, without seeing any value coming out of this project.

Does the Agile Manifesto have anything to offer here? Yes, of course: Responding to change over following a plan. The basic principles and techniques of the agile framework are built to adapt to the changes with minimal or limited cost/schedule implication.

In a fixed-price project, it is in the best interests of both the customer and provider to keep the schedule and cost intact, although changes are inevitable. The agile techniques allow the project manager to collaborate with the customer to modify the change request to “exchange request.” “Exchange requests” work like this: each time the customer and I need to change the specification, we make a change request and estimate its effort (and thus cost). When the customer approves the change request and its estimate, he or she can add the new features to the project if he or she first removes a functionality requiring at least the same effort. We can only remove unimplemented features that both the customer and provider to keep the schedule and cost intact, although changes are inevitable. The agile techniques allow the project manager to collaborate with the customer to modify the change request to “exchange request.” “Exchange requests” work like this: each time the customer and I need to change the specification, we make a change request and estimate its effort (and thus cost). When the customer approves the change request and its estimate, he or she can add the new features to the project if he or she first removes a functionality requiring at least the same effort. We can only remove unimplemented features that the development team is not working on. For example, the customer can add feature X (which costs 5 person-days) if he or she first removes feature A (which costs 3 person-days) and feature B (which costs 2 person-days). What are the advantages of this technique?

• We keep the budget and timing constant (the definition of a fixed-price project). The development team and the customer have the satisfaction of finishing a job on time, on budget.
• We are able to change the specification flexibly to deliver what the customer needs, not necessarily what he or she asked for (2 years ago).
• We do not deliver what was specified, but we do (at least) as much work as agreed on. If we have revised the specifications, the new features are more valuable than the old ones or the customer would not have swapped them in; this means that we deliver something that’s at least as valuable as what was agreed on.
• We put more thought into changes to the specifications: the customer has to think very carefully if the new feature is really worth more than the feature being taken out; therefore, we expect fewer, but more useful, changes to the project.
• We shorten the feedback loop between functional changes and their effects on budget and timing, so that they are immediately visible.
• The customer’s project manager becomes more responsible for budget and timing.

Keep in mind that the swap was easily possible, because the development team had not put any effort into the features that were swapped, so both the project team and customer are in a win-win situation. Now, what happened to the dropped
The dropped features can be added to the wishlist, which can be used in a follow-up delivery. Don’t worry that these are lost money; rather, more features that will be added to those dropped features after the end user really starts using all the features provided.

From a project manager’s perspective, he or she would like to see the changes coming in the earlier part of the project, because he or she will have more time to react to those changes. If the change was originated based on the visualization or utilization of the system, then agile development techniques cultivate those change requests to surface during an earlier period of the project life cycle by providing working software to implement or provide feedback.

What is the Catch?
It seems like if we introduce agility to a fixed-price project, everyone will be happy:

• The customer will be happy because the project was completed on budget and on time.
• The customer will be happy because he or she was able to introduce changes that were more important to him or her in the middle of a fixed-price project and he or she got what he or she really wanted.
• The end user will be happy because he or she can use the system (at least the crucial features) earlier than originally anticipated.
• The project manager will be happy because he or she was able to add another successful project to his or her portfolio.
• The sales manager or finance manager will be happy because there is potential for a follow-up project, because we dropped a few originally requested features and he or she has a happy customer.

The above-mentioned items represent an ideal situation and cannot be realized without some serious cultural changes.

• The most important requirement is that there should be a constructive, honest, and trusting working relationship between the customer and provider. It takes a lot of time and effort to build up this relationship: trust must be earned. The best way the provider can earn this trust is to be honest and deliver on his or her promises.

Strangely enough, few customers are prepared to put this amount of work into their important projects. They think they can let the provider do most of the work. It should be clear that a software project can only succeed if both parties do their parts of the job.

Let me put myself in the customer’s shoes for the following scenario. My wife and I decide to custom build our dream home, and yes, it is our first home. We do not work in the construction field and barely understand the civil engineering drawings and specifications. After enough hunting for builders, we short-listed two of them, who propose their procedures, as described below:

Builder 1: You will have to enter into a fixed-price contract with me, which will protect you from the price fluctuations of the building materials. The price will be fixed based on the various materials that you choose to build the house, as well as the various amenities selected. The detailed specifications will be provided to you 2 months from the day you sign the contract. We will provide 2 weeks to provide feedback on the specifications; after that period, you will not be allowed to visit the site for another 5 months. No changes, even minor ones, will be accepted after the review period. Seven months from the contract date, we will allow you to visit the home, conduct an inspection, and do a walk through. The house will be under full warranty for 1 year.

Builder 2: You will have to enter into a fixed-price contract with me, which will protect you from the price fluctuations of the building materials. The price will be fixed based on the various materials that you choose to build the house, as well as the various amenities selected. You are most welcome to visit the work site when you wish, but on a monthly basis, we will conduct a formal walk through with you to demonstrate the progress of the construction. During that time, we will spend time discussing our plan for the next month; if, indeed, you want to swap any of the next month’s activities with one of the future activities, we can do that, provided it is architecturally feasible. In addition, we provide you with the flexibility to alter your decisions on the amenities yet to be built, provided it is architecturally feasible and will not add more effort to the previously
finished activities. However, if the change requested does affect the cost, we will have to work together and perhaps swap them with some of the unimportant amenities. Please remember that we will not compromise on the structural quality of the building and will not accept any requests that could adversely affect the quality. We will provide full warranty on the house for 1 year.

Given the above situations, which builder will I choose, knowing that, my wife, my uncle, my dad, and my friends are my main advisors and I, myself, will gain many ideas during the course of the construction of our dream home?

Conclusion
Although agile software development methodology does not offer any silver bullet to solving all the issues related to fixed-price projects, it offers tools and techniques that could be used to add agility to the fixed-price projects. More than the type of the contract, the success of the agile techniques will depend on:
• Delivery structure proposed during the selling phase
• Culture to foster customer collaboration
• Commitment to deliver high-quality product, which will satisfy customer needs

References

Web References
Agile Fixed Price Projects by Pascal Van CauwenberghNayima
http://www.pmhut.com/the-chaos-report-2009-on-it-project-failure
http://agilesoftwaredevelopment.com/blog/peterstev/10-agile-contracts

About the Author
Siju P. Varghese is a manager with Deloitte Consulting LLP. He has over 12 years of consulting experience and over 3 years of experience in managing multiple high-visible, high-value initiatives in the Department of Public Welfare for the State of Pennsylvania and the National Finance Center in the federal domain.