Time is money – agile fixed price

Faced with having to deliver a fixed price, fixed scope project, what development methodology should we follow to maximize our chance of success? Richo Strydom, CTO of Valtech Ltd, offers his perspective on the most risk-averse method of delivering business-centric software.

They say that you cannot have your cake and eat it. Unfortunately many IT procurement processes still assume you can. Fixed price, fixed scope development contracts are still prevalent in today's marketplace and many of us will be involved in delivering these for some time to come. Typically this is a commercial decision taken once it is clear that this is the only way the client is willing to contract.

This reality leaves us with an important question. Given the task to deliver a fixed price, fixed scope project, what development methodology should we follow to maximize our chance of success?

Some might argue the **waterfall** process due to its rigorous focus on upfront planning, tracking and documentation. But extensive research indicates that the waterfall is in fact not appropriate for any but the simplest projects irrespective of how they are contracted commercially. This is mainly due to the lack of useful feedback, i.e. built and tested software, until very late in the project, when it is very costly to correct mistakes in requirements or architecture and too late to improve the process.

Iterative and **evolutionary** development aims to address this shortcoming. These are risk-driven and focused on continuous measurement and improvement. For fixed price projects, where late delivery, misinterpreted requirements or last-minute architectural surprises directly translate into budget overrun, doesn't it make sense to utilize a process that puts early risk mitigation at its core?

Time is money. Especially in fixed price projects where any time not spent on producing useful deliverables reduces your margin and contingency and increases delivery risk.

Time is money. Especially in fixed price projects where any time not spent on producing useful deliverables reduces your margin and contingency and increases delivery risk

Agile methods, a subset of iterative and evolutionary methods, aim to reduce waste and focus effort where it matters most in software development: working, fit for purpose software and the artefacts required to create, deploy, operate and maintain it. Agile methods are therefore an attractive option when you need to reduce wasteful overheads.

Unfortunately many advocates of agile methods still believe that agile is only applicable to time and material (T&M) style projects and little attention is paid to the reality of how to run a project where T&M is not an option. I believe that agile, iterative and evolutionary practices, when carefully applied, are indeed an appropriate approach to the delivery of fixed price, fixed scope projects.

Agile methods

I will use the Manifesto for Agile Software Development, which summarizes the core values common to agile methods, to explore the areas of agile that need special attention in the fixed price, fixed scope context.

Individuals and interaction over process and tools

Agile methods encourage face-to-face communication, removing barriers to understanding of complex requirements and concepts. However this does not mean all communication should be informal.

Experience has shown that verbal communication alone is not adequate and more persistent mechanisms, even if applied informally, are required to ensure information is available to all team members at all times. Valtech has found that lightweight collaboration tools, e.g. a Wiki, are invaluable to complement face-to-face communication.

The aim must be to provide just enough process and tools to make the information generated through discussion and personal interaction available to all stakeholders.

Working software over comprehensive documentation

Agile methods aim to reduce waste. It is a continuous process, constantly evaluating and streamlining the process, removing any activity not directly adding value.

So what is waste? Newcomers to agile often argue that the end goal is working software, hence anything other than writing software is waste. Even though working software is by far the most visible deliverable, it is not the only deliverable required for a successful software system.

Additional deliverables differ from project to project but usually take the form of documentation. Documents do not have to be overly formal and rigorous sign-off practices are actively discouraged. Lightweight, fit-for-purpose mechanisms to capture information at the source throughout the whole development cycle is ideal. These tend to encourage all stakeholders to contribute and allow the required end documentation to evolve to the appropriate level of detail.

Ironically embracing change is simultaneously one of the most useful and dangerous aspects of the agile philosophy and requires discipline from both delivery team and customer to ensure the project vision is not lost

Counter-intuitively this approach often produces more complete documentation, since continuous lightweight information capture tends to keep information more up-to-date than formal document management processes.

Customer collaboration over contract negotiation

Not surprisingly this is one of the main areas where the commercial reality of fixed price, fixed scope contracts are at odds with the agile philosophy, and the contractual arrangement should strike a balance between the commercial needs and the benefits of an agile approach.

Research shows that clients are likely not to know what they need at the start of a project and in many cases initial requirements are high-level and better described as scope statements.

Agile methods accept this reality and encourage collaboration between the delivery team and the business to elaborate the requirements during the project, providing feedback through useable software that in turn feeds the requirements process.

In fixed price projects this process should be carefully managed. Depending on the contract, contingency and maturity of the customer relationship one might allow requirements to be refined to a certain extent within the defined scope but shouldn't allow scope to change without appropriate change control. There is a significant difference between scope and requirements change. Scope identifies the general features and size of the project whereas requirements are the detail description of how the scope should be realized.

A lightweight and efficient change control mechanism is encouraged where change is agreed and audited between an empowered customer representative and the team project manager. The change control process should form part of the contract so that it is clear from the start what is expected from each project participant.

Respond to changeover following a plan

Agile methods encourage change in order to improve the success of a project, whether due to requirements change or driven from technical feedback, testing, third party dependencies etc.

Ironically embracing change is simultaneously one of the most useful and dangerous aspects of the agile philosophy and requires discipline from both delivery team and customer to ensure the project vision is not lost. Iterative development's focus on the short-term iteration objectives has a hidden danger in that it is attractive not to worry too much about higher-level progress. It is therefore important that change be actively monitored, even if just to take note of knock-on effects to project milestones or third party dependencies.

A common misinterpretation of the agile manifesto is that all planning and tracking is pointless since it is all going to change in the future. This is far from the truth. High-level planning and even estimation can be very useful even in smaller projects but becomes essential in the larger ones.

A good analogy is considering a journey to an unfamiliar destination. One approach is to start driving hoping to find directions along the way. You might reach your destination but it is unlikely you would have taken the most direct or fastest route. Another approach might be to consult a map, traffic reports and other useful information, and pick waypoints along the way. When for some unexpected reason a part of your route is no longer viable, or your requirements change and you decide to take a scenic detour or even choose a different destination, you can adjust your initial plan and pick a new route for the rest of your journey.

This is the essence of agile planning. You accept that things are likely to change, you only plan high-level milestones to keep track of the big picture and focus on the detail only for the foreseeable future. When there is a requirement for change, consider whether this is in line with the vision and intent of the project, and if so manage it appropriately.

It is therefore important that the whole team understands the intent, scope and commercial model of the project and the importance of keeping track of change. Technical and requirements change should be noted so that others in the team are aware of possible impact. A lightweight mechanism is more than adequate as long as it provides history and easy access to all members.

My final thought

In an ideal world there would be no such thing as fixed price, fixed cost projects. One day this may be true, when all businesses understand the benefit of a more flexible and adaptive commercial procurement model.

Until such time however, here at Valtech we believe that careful application of modern agile, iterative and evolutionary best practice is the most risk-averse method of delivering business-centric software. ■

Iterative and evolutionary methods

Extensive research has shown that the following best practices significantly increase the success of software projects independent of the commercial model. These are all at odds with waterfall development and common to all iterative and evolutionary methods:

- **short time boxed iterations:** analyze, design, build, integrate and test in every iteration;
- **risk driven development:** focus on both the technical and business risk areas first;
- **constant feedback:** improvement of both the software and the process;
- **integrate and test frequently:** unit tests as well as functional end-to-end tests;
- **short-term project management focus:** detail task identification and tracking within the current iteration, and milestone planning at the project level.

Richo Strydom is CTO of Valtech Ltd, a global IT consultancy and solutions provider. For further information please contact Valtech on tel: (020) 7014 0940, email: richo.strydom@valtech.co.uk or vist www.valtech.co.uk