Selling Agile to the CFO: A Guide for Development Teams

You've learned about agile development, or perhaps you have even worked in an agile organization and have now moved to a traditional one. You're convinced that by adopting agile, your team deliver better value to your business, more frequently, and at a sustainable pace. [Hendrickson, 2009] You know that adopting agile will take a big investment by your company. Your team will need time to master agile values, principles and practices, and apply them to your unique situation. A culture of learning, time to experiment, and tolerance for mistakes are a must to successfully improve software quality and respond to changing business requirements. How can you get the resources and long-term support you need from the business?

Here's a good strategy: Get your Chief Financial Officer (CFO) on board. Telling the CFO "We want to do this" probably won't work. How can a developer convince a CFO that agile development is worth the investment? After speaking with both financial and software experts, including my own company's CFO, I'm sure that you can make agile an easy sell to your CFO.

Consider the CFO

Before you meet with your company's CFO, learn about her roles and responsibilities. CFOs project financial performance and try to predict the return on potential investments, but they often make key contributions outside of finance. Your CFO may know a lot about all aspects of the business, and may know be more interested in software development than you think.

CFOs are responsible for budgets and controls. They need measurability, transparency, and benchmarks to help plan for and predict future performance. They're routinely asked to invest in improvements. Be prepared to show how agile provides an ability to quantify software development where traditional methodologies don't.

Prepare to Be a Change Agent

Change is hard, but necessary for organizations to improve and innovate. Each of us can help effect change, such as a transition to agile. In addition to preparing information to educate the CFO about agile, learn good ways to introduce new ideas. The book *Fearless Change: Patterns for Introducing New Ideas* is one way you can improve your chances for success. For guidance on your meeting with the CFO, check out the pattern "Whisper in the General's Ear" [Manns/Rising p. 248-9].

Quantify Agile Development Advantages

CFOs need benchmarks, tracking, and financial measurements. Explain how agile delivers value frequently, in small increments. Incremental development aids in

gathering metrics that facilitate budgeting, efficiency analysis and help the bottom line. Go over the process of breaking features into small user stories and sizing them with a point system. Over time, CFOs can get a good idea of how much a story point costs. As teams learn, their velocity will average out and become predictable.

Describe how agile teams build only what is useful and necessary. You can compare this to just-in-time inventory and the drop shop model for online orders: build features in shorter timeframes and know what you're getting. CFOs know that long development cycles are inefficient. The short cycles of agile, delivering production-ready software in one, two, three or four weeks, provide more certainty. You know what you need to do in the next month. CFOs understand this much better than waterfall development. If software is your company's most important product, this just-in-time approach is especially important.

Quantify Technical Debt

Many, if not most, traditional software projects (and some so-called "agile" ones) are dragged down by technical debt. If your code is already clean and continually refactored, going into 'debt' by cutting some corners to meet a critical business deadline and 'paying it back' with later refactoring is a sensible approach. If your code is a nightmare of brittle, mysterious code that isn't supported with automated regression tests, cutting corners even more leads to disaster. Even the smallest code change might result in dozens of regression bugs, paralyzing the project.

Managing technical debt is a compelling justification for adopting agile. A whole-team and whole-company commitment to quality along with agile practices such as refactoring, TDD, ATDD and CI helps teams start chipping away at their technical debt.

The ability to quantify technical debt, according to my company's co-founder and CFO, Dan Gutrich, is music to a CFO's ears. In "The Financial Implications of Technical Debt" [jimhismith.com, 10/19/2010], Jim Highsmith points out that the impact of technical debt is often slow-growing and hidden. CFOs understand the concept that technical debt increases the cost of change over time, so be prepared to explain how an investment in reducing technical debt will pay off with consistent, predictable delivery of value over time. Agile practices such as refactoring and TDD help manage technical debt and allow the team to sustain a maintainable pace as they deliver value frequently.

Israel Gat explains how to calculate your current level of technical debt in the October 2010 Cutter IT Journal issue on Technical Debt. His process starts with code analysis and identifying quality deficits, then figuring the time and money needed to fix the deficits. He states, "Once technical debt is monetized, its operational, financial, and business implications can be understood and appreciated by any stakeholder. A CFO will have no problem relating to a statement like "the technical debt on the project amounts to $500,000."

Technical debt piles up from a test as well as a code perspective. Ken Pugh defines Acceptance Test Debt as "the nonexistence or non-automation of acceptance tests. Developing product features is more risky when an organization has acceptance test debt. The debt corresponds inversely to the proportion of requirements that are covered by automated acceptance tests. To reduce the risk, you need to decrease the debt." ["The Risks of Acceptance Test Debt", Cutter IT Journal, October 2010] He suggests quantifying this debt by establishing the percentage of requirements covered by acceptance tests, and the percentage of those tests that are automated. Low coverage for either manual or automated tests represents high debt.

Lisa Crispin 12/19/10 11:34 AM

Comment: This section on technical debt seems long – maybe take out this paragraph and just mention elsewhere that tech debt applies to both tests and code?

Do your homework, learn how to quantify your technical debt and present the results to your CFO. Be ready to discuss why higher quality and fewer bugs directly affects the bottom line. Should your company invest time and money to reduce technical debt, versus spending lots of time and money to fix production bugs while new development is slowed? It's a simple choice for most CFOs.

Transparency

In organizations that practice phased-and-gated development, CxOs live in blissful ignorance of software project progress until the release date, when they learn that the release has been postponed. In agile organizations, the CxOs may sometimes experience the panic of looking at a task board mid-sprint that has too few cards moved to the 'done' column – but they much prefer that to unpleasant last-minute surprises.

The difference between agile and traditional projects is dramatic from the CFO point of view. In a traditional project, with long release cycles, there's no way to know the status of the project on a daily basis. Project managers might have Gantt charts or project plans, but these don't accurately predict the final release. The many sources of immediate feedback on an agile project provide CFOs with much more certainty and predictability.

Demonstrate agile's many "big visible charts" to your CFO. Show how the project's current status will be visible in many ways: release and sprint burndown charts, task or Kanban boards showing what's done, what's in progress, and what's blocked, results from continuous integration. Even if an iteration goes awry and some user stories are not completed, the reasons can be identified immediately with retrospectives, and the project can get right back on track.

Continuous integration is an 'agile' practice that's essential for any team. CFOs understand that the number of tests at the unit, API and GUI level should increase over time, and that they should all pass. In my team's early days of practicing agile, our managers got reports of daily build success. If tests failed two days in a row, our CFO noticed and asked about it. Quick, visible feedback lets the CFO know projects are on track.

Explain to your CFO that the development teams will collaborate closely with business experts, gaining much domain knowledge. They'll be much more likely to understand customer needs for each user story and feature. This reduces churn and helps the team find better software solutions.

The idea of limiting work in process, and ensuring each story is 'done' (including testing activities) before moving on to the next story should make sense to any CFO. We can't know for sure how much work is left to do, but we know what we've finished.

Some years ago, I worked for an internet retail company who wanted to adopt agile, but never had enough motivation to try. Then a seemingly-impossible project with a set-in-stone deadline appeared. The CTO asked me what in the 'agile' world could help us succeed, and we decided to use a Scrum approach. The customers came up with a huge list of stories that had to be done by the initial release. When I asked the customer to choose between having 100% of the features 80% done, so that none could be used, versus 80% of the features 100% done and ready to go, with the remaining 20% handled offline, the choice was obvious. The 80% was done and working by the deadline, and the project was a success. CFOs understand these choices.

Lisa Crispin 12/19/10 11:35 AM

Comment: Maybe cut this – it seems wordy, I could probably make the point without the story about it?

Trust

Many of us have worked in companies where speed was valued over software quality, and development teams were subjected to "death marches" to kick software out the door. It's easy to become a bit paranoid, but according to our CFO, Dan, CFOs don't have an innate distrust of technical teams. You may find your CFO is receptive to your explanation of how agile development will help your company's bottom line.

An Additional Sales Pitch

When my company adopted agile development (a decision by our CEO and CFO), agile was relatively rare, practiced by only a few teams. Now agile is mainstream, practiced by many companies of all sizes and in all industries. As Dan notes, this means that practicing agile has become a recruiting and retention tool for Human Resources. Just as CFOs are aware of incentives to attract and retain salespeople, they want to get the best developers as well. The ability to recruit the most talented developers is another good reason for adopting agile principles and practices.

Long-Term Success

Too many agile transitions start out with a lot of enthusiasm, but die on the vine. It takes many months, even more than a year, to get traction on practices such as driving development with tests. Programmers, testers, business analysts, project

managers, everyone involved with development and their customers have a lot to learn.

Plan for feedback throughout the process, and hold retrospectives at frequent intervals (at least every iteration). Dan suggests looking back over the first year of using agile development to gauge the return on investment. When my own current team tallied up the results of our first agile year, we found we had delivered even more story points than planned, and the production defect rate was dramatically reduced. Given that we focused solely on quality, not speed, and often struggled as we learned techniques such as TDD and refactoring, these results surprised us, and delighted our CFO.

Your agile transition won't always be smooth sailing. Even as the years pass, new challenges will arise. By this time, you'll have built trust with your CFO and business experts so that they will continue their support. Don't take anything for granted. Keep using appropriate measurements and quick, visible feedback to guide your continual improvement.

Analysis and numbers, long-term benefits and efficiency gains that translate to money – these are the compelling arguments to present to your CFO when you want your employer to commit to agile development. The twelve principles behind the Agile Manifesto [http://www.agilemanifesto.org] are a great place to start. What CFO wouldn't want to satisfy the customer through early and continuous delivery of valuable software? Using working software as the primary measure of progress makes good sense to financial and business experts. Let the agile values of collaboration and communication guide you as you discuss agile with your CFO and build mutual trust.

References and further reading:

The Financial Implications of Technical Debt, http://www.jimhighsmith.com/2010/10/19/the-financial-implications-of-technical-debt/, Jim Highsmith, 2009

SpamCast 112, Israel Gat, Technical Debt, http://www.spamcast.libsyn.com/s-pa-mcast-112-israel-gat-technical-debt, December 2010.

Cutter IT Journal issue on Technical Debt, free download with registration: http://www.cutter.com/offers/technicaldebt.html

Fearless Change: Patterns for Introducing New Ideas, Mary Lynn Manns and Linda Rising, Addison-Wesley, 2005.

http://testobsessed.com/2009/05/26/defining-agile-results-characteristics-practices/, Elisabeth Hendrickson, 2009

Acknowledgements:

Many thanks to Dan Gutrich and Pat Reed for their help with this article.