"A Practical Approach to Large-Scale Agile Development", ISBN 0321821726

Chapter 2

Tuning Agile to Your Business Objectives

We started with the "why" of agile practices—that is, principles. But there's a deeper "why" that we need to explore. It's really the "why" of the principles you choose to follow. Although agile is a powerful concept, becoming agile just because "it's the thing to do" won't automatically help a business achieve what it is trying to accomplish. To successfully create the significant breakthroughs in your development effectiveness that are possible with agile, it has to be aligned with *why you want to do it* in the first place and *what you need to achieve from it*. You should be agile not just to be agile, but to drive the business results. Start by describing your business situation.

First, sit down and identify your current business realities (where the money and time is going) and strategic objectives (where the money would ideally be spent for your business situation):

- **Cost and cycle-time drivers**
  What are the activities that are consuming your resources and limiting your ability to deliver on time?
- **Value proposition**
  What are your products or services really trying to achieve for the customer?

"A Practical Approach to Large-Scale Agile Development", ISBN 0321821726

The final step in establishing the backdrop for an agile transformation and making sure the efforts are tied to your real business needs is to combine the two lists (where are you investing now; where do you need to invest) for a clear view into the problem areas. Use this analysis to *establish clear development objectives* for your organization. *The biggest cost drivers that aren't key to the value proposition are targets for improvements.* If these cost drivers can be architected out of the system, automated, or engineered away, it can free up resources for innovations critical to the value proposition.

The best way to talk about how we tuned agile to our business objectives is to clearly spell out our business situation before our large-scale agile experience began. So in this chapter, we'll give you an overview of HP FutureSmart Firmware that we're using as the case study. We'll identify our costs and cycle-time drivers prior to our agile transformation, explain the value proposition our business needed, and then list the development objectives that came out of our analysis and would effectively close the gap we faced.

Background: HP FutureSmart Firmware Case Study

HP FutureSmart Firmware is the name the business uses to market the latest embedded code used to control LaserJet hardware and enable solutions resident on the device. A typical laser printer consists of the electromechanical print engine, which is controlled by a formatter. The formatter is made up of both electronics and logic. The logic is referred to as *firmware* but can be considered a full-on multitasking operating system. In this case of the firmware for enterprise-class printers and copiers (FutureSmart), it is as complex as the operating system and logic running your PC or laptop or smartphone.

Our business challenges started with a predicament of two-year long development cycles for delivering firmware, and of complex embedded software that had been slowly aging over many years and needed to be

"A Practical Approach to Large-Scale Agile Development", ISBN 0321821726

re-architected. Big-bang integrations were frequent. Before learning about agile, we had some early improvements and got to the point of 8-week development cycles, a daily build or two, and a nightly smoke test. But even with these significant improvements, some significant inefficiencies existed.

Cost and Cycle-Time Drivers Prior to HP FutureSmart Firmware

The first step is to understand how resources are being deployed and what activities are driving development costs. Honestly assess where software development dollars are spent. It is also important to understand the cycle time for a developer to implement a change and then get feedback on if it works.

Throughout this experience, we've had around 400 developers worldwide needing to get firmware and test changes integrated into a firmware system consisting of several million lines of code, with very high quality expectations. When we started our transition to agile development, we had created a complex environment and code base over many years that took most of our efforts just to keep it going:

- Ten percent of our staffing was for "build bosses" (someone on each team designated as its full-time code integrator) plus a central integration team to accomplish the one or two builds per day we were doing, with many teams doing integration their own way. This was a very manual process of integrating and reverting code, consuming several highly qualified engineers who, as a result, spent very little time actually coding. In this environment, each project team would have a build boss that would gather all the changes by the team every few days and bundle them into a collection of changes. These changes would then be provided to the integration team that would be taking changes from 15 different build bosses for a nightly build with a smoke test. This nightly build would then be provided for additional testing over time.

"A Practical Approach to Large-Scale Agile Development", ISBN 0321821726

This approach resulted in a resource sink, but more importantly, it could be up to a week from the time a developer made a change until it got into broader testing on the main code branch to see if it worked.

- Twenty percent of our resources were spent doing detailed planning for future feature commitments that quickly became obsolete or were never delivered. Business and marketing expected a clear "final list of features" one year before product introduction. To provide that commitment, we worked on detailed work breakdowns, schedules, integration plans, and estimates, all of which required constant maintenance and revision, because new discoveries and adjustments are an integral part of any high-tech research and development (R&D) effort.
- Twenty-five percent of our resources were consumed porting the existing codebase and features from one product to another. Because of schedule pressures, we hadn't spent sufficient time to abstract out the code and encapsulate product differences. We also ended up splitting the organization and creating three distinct branches of the previously common code. This meant more focus for each part of the business, but fewer resources for assuring code maintainability.
- Fifteen percent of our development costs were for manual test execution, which was a significant cost driver. Although we had a very large test suite, most of it needed to be executed by technicians, which was a large chunk of the budget. It also meant very long feedback loops from test to development. It was sometimes weeks or even months between when a firmware change was made and when a test actually found an issue. This made for long find/fix cycles, and it consumed a large chunk of the budget. This also meant that we frequently could not add products to our plans because we did not have the resources for testing them.
- Twenty-five percent of development resources were deployed supporting existing products, either fixing customer change requests or making sure we had a consistent set of features across printers

and multifunction products (MFPs). With a focus over many years on getting each product to market, we had created multiple code branches that all had to be maintained for the products in the field.

- This left us with limited capacity to focus on the value proposition and customer differentiation that would actually provide the business value needed for continued success.

So that gives a clear picture of where we were allocating our resources. If you add it all up, the firmware development cost drivers were 95% "get the basic product out" and just 5% adding innovation. That is exactly opposite of where the business needed us to be in order to be competitive in the marketplace. So where did we *want* to be spending our money? What was the business asking for? The next section describes our value proposition for making such a substantial change.

Value Proposition of Re-Architecting the HP FutureSmart Firmware and Processes

After establishing where you are spending your resources, it is important to clearly understand the value proposition of your product. Is your primary goal to reduce cost for a given functionality? Is it to release the largest number of products at a given cost? Or is the real opportunity to provide clear differentiation to the customer? Each of these is valid, and there are many more, but the decisions around the trade-offs to make in transitioning your development processes will be dramatically different depending on your specific business value proposition and cost drivers.

To start our agile change, we stepped back and asked what we really wanted to accomplish. What if we could change that "95% turn the crank" reality into something very different with innovation at our core? Following are the real business drivers that we established as our vision and value proposition:

"A Practical Approach to Large-Scale Agile Development", ISBN 0321821726

- Our firmware had been on critical path for nearly every product delivered for more than 20 years. We sorely needed to get it off that critical path. How could we *deliver firmware early and often with even higher quality*?
- Because such a large percentage of our resources was spent on the "turn the crank" activities mentioned previously, a significant pent-up market demand existed for more features and innovation. For four years, we tried to spend our way out of the problem, increasing firmware R&D investment dollars by two and a half times across multiple versions of the code base that had split off in an attempt to let each business unit control its own destiny. But it didn't seem to help. We needed to *significantly improve developer productivity and organization agility* to truly lead the market in all desired product attributes and features. We needed to engineer a solution.
- Our business drivers were also changing. Customers had been moving from a previous focus on "buying up to the latest product for faster printing" to needing an *advanced and consistent set of Multi-Function Printer (MFP) features for workflows/solutions* that have the MFP as an integral part. Previously, it had been okay for different products to have different capabilities. But after MFPs became an integral part of their workflows, customers started demanding consistency in the feature set. The technology curve with printers was to the point where the hardware engine speeds and print quality were satisfying customers, and we didn't need to keep ramping up the curve of higher speed or print resolution. More and more product differentiation began originating in firmware. Our firmware had transitioned from a "thin layer of code to help control the print engine" to being more software-like as the critical enabler for supported workflows and solutions. Customers also began to manage their fleet of printing devices, raising the importance of managing devices in a consistent manner.

With these clear cost drivers and opportunities as our backdrop, we were prepared to put agile to work for us (not us for it) and tackle these difficult but critical disconnects in our investment versus value-add picture.

Establish Development Objectives from the Business Analysis

As we started out on our agile journey, we translated this business picture of "where we were spending our money" and "what the business needed" into a clear set of **firmware development objectives** that we felt would close the gap between where we were and where we needed to be. Our goal was that these objectives would help *unleash the product roadmap and enable innovation by reengineering the code and development processes*:

- Create a stable application code base that is always close to ready for release.
- Automate tests and run a full set of regression tests every night.
- Automate the integration process, including autoreverting any code that is not up to par.
- Significantly reduce the work needed to get new products working with high quality and out the door to the market.
- Re-architect to remove product differences, enabling one branch for all products (even refreshes of released products).
- Improve developer productivity by a factor of 10 (build times, streamlined processes).
- Create a common development environment so engineers can easily help across teams.
- Reset expectations and reduce feature estimation activities (commit by delivering).

"A Practical Approach to Large-Scale Agile Development", ISBN 0321821726

There was a final consideration in determining how to go forward: Every team or business must not only step back and ask about resource allocation and value-add, but also about the capacity of the organization to absorb change. How much change can the organization handle, how fast, and how many ideas can be driven from your position in the organization? This example involves a large amount of change over a long time period with a big team. Agile transformation can only happen as quickly as your organization has the capacity to invest and is ready to embrace significant change along with that investment. It also matters how influential the thought leaders are in your organization. The final chapter goes much more into the best way to start, but no matter what, make sure you start with the items that will give the biggest bang for the buck and are appropriate for your organizational influence and leverage.

In the following chapters, we will share our experiences and changes that enabled the following results in hopes that it will inspire your organization to start your journey toward transforming your business:

- 2008 to present overall development costs reduced by 40%
- Number of programs under development increased by 140%
- Development costs per program down 78%
- Firmware resources now driving innovation increased by a factor of 8 (from 5% working on new features to 40%)

Summary

What are your business objectives and value proposition? How are you spending your time and resources? What are the pain points you want to overcome? Do your investment areas match your objectives? With our case study as an example, we encourage you to do the same exercise. This will allow you to create a vision for the organization and a roadmap of how to get there. It's easy to get lost in the day-to-day and sprint-to-sprint nature of agile. Having the vision and strategy for where you want to be in the medium or long term is a powerful backdrop to measure all activities against and know if you're being successful.