1

The Persistence of Waterfall Planning

The benefits of increased business agility have been well documented by the Agile community. Who would argue that organizations that can rapidly react to changing market conditions will prevail? Or the potential for increased R&D Return on Investment (ROI) using the "Minimal Marketable Feature Set" (MMFS) approach to provide value at lower R&D cost? Yet, most organizations constrain Agile development with Waterfall planning practices and large releases with fixed content and schedule.

We'll start with a fictional startup accounting company called AccuWiz that used Agile development to become successful. As the business grew, they turned into a company many of you may recognize if you work for a large company. AccuWiz tossed aside the Agile planning and development practices that made them successful and became like companies from which their Agile developers fled. For those of you in smaller companies, this is the fate that awaits you as your company grows from entrepreneurship to a product organization expected to set and meet financial commitments.

Although these changes were mysterious to the developers, we'll see that there are natural business forces that explain the changes. The example demonstrates how Waterfall planning will continue to constrain Agile development unless ways are found to satisfy the business need for predictability.

AccuWiz will be revisited in a later chapter to see how they were able to use the Investment approach to restore them to an innovative organization with highly motivated Agile teams, while meeting the predictability needs of the business. There is hope for your organization.

1.1 Introduction to AccuWiz

AccuWiz was a startup that leveraged Agile development to get traction in the crowded accounting software market. The founders, Ken and Brett, brought in a

Unlocking Agile's Missed Potential, First Edition. Robert Webber.

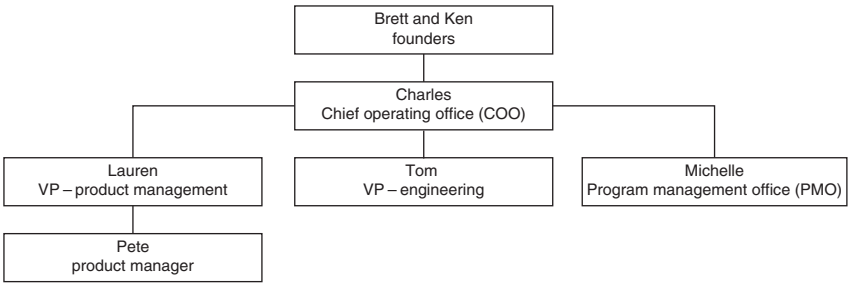

Figure 1.1 AccuWiz organization chart.

new chief operating officer (COO), Charles, to establish financial predictability to support their goal for acquisition or a public offering.

As companies grow, departments evolve that have different, and even conflicting, objectives. AccuWiz adopted the typical product development organization (see Figure 1.1).

1.1.1 The New COO

Charles, the new COO, was the VP of product management of a large established company that has dominated the accounting software market for years. Charles was brought in to establish the financial predictability sought by the founders. There is no way AccuWiz can be acquired or go public without a record of financial predictability.

Charles created a product management organization and hired Lauren from his previous company to lead the department. Lauren is expected to implement roadmap planning to support AccuWiz financial growth objectives. Lauren hired product managers with years of experience in the accounting field. Pete, for example, is a trained accountant who developed a keen interest in software by working with several traditional accounting systems. He can't wait to share his experience with software developers. He has so many great ideas. He especially likes the idea of being the "Voice of the Customer."

Tom was the first technical person hired by the founders and rose to the position of VP. He built his development organization around Agile development. Tom and his team provided many of the technical innovations that differentiated AccuWiz.

A new program management office (PMO) headed by Michelle was formed. Charles brought her in from his prior company because of her reputation for holding people accountable. She is supposed to improve software project schedule and cost predictability.

The founders, Brett and Ken, feel they have established the organizational infrastructure to take them to the next level.

1.1.2 Product Management

The new product management department brought structure into the planning process. They organized the hundreds of feature requests from sales into a prioritized list and allocated them to annual releases on a three-year roadmap. Lauren pushed Tom to commit to as many features as possible in each release.

However, Pete, the product manager, has become frustrated. This isn't the dream job he expected. He is under intense pressure from his boss, Lauren. It seems that all Pete does is pass on feature requests from sales to development. The VP of sales often goes directly to the founders to complain that they will miss their revenue commitments next year without these new features. Sales has a lot of influence with the founders, especially through their weekly golf outings. Pete has had to put all his great ideas on hold and can hardly keep up providing requirements for the myriad feature requests.

Pete must pressure developers to reduce estimates. His boss, Lauren, is very unhappy when he comes back from a meeting with engineering where they refuse to commit to a feature in the next release. She tells Pete to keep pressing. She knows they have slack. Whenever they apply pressure, engineering finally accepts the features.

When Tom balks at including features in a release, Lauren often sets up a meeting with Tom, Charles, and the VP of sales to emphasize the importance and potential impact on revenue. Tom is in the "hot seat" and usually commits with no relief in schedule or feature commitments.

1.1.3 PMO

Agile development was new to Michelle. Tom told her that releases must have feature content flexibility in Agile because requirements are created and changed throughout the release cycle. He also tried to explain the natural uncertainty in software estimation, but Michelle feels this is just a wishy-washy way to avoid commitment. Tom also explained that the teams must incorporate feedback during development to deliver what customers really want. Tom says, "The business needs to be more Agile."

Michelle has no idea how she is supposed to meet her roadmap commitments. There is no schedule or feature content flexibility. The schedules were established without any contingency, and any changes will impact financial commitments. And she and Charles know from their prior company that contingency reduces pressure on development, and they slack off. She has already observed that there are very few developers in the office when she comes in at 8:00 a.m. and has passed the observation on to Charles.

Michelle took an introductory Agile class for management to try to get on the same page as Tom. However, there was nothing about how multiyear business

roadmap and financial commitments can be planned and achieved with any predictability. She heard the same expression used by Tom – "The business needs to be more Agile." She can't imagine Charles telling the founders they have to invest millions over the next three years, but he can't commit to financial returns because sales doesn't know what they'll have to sell. "Just be more agile." Charles wouldn't be around long.

1.1.4 Engineering

Tom, the VP of engineering, has been with the founders from the start. It was his idea to use workflow to differentiate their product in the market. Tom chose Agile development to quickly respond to individual customer needs. He soon found that versioning became a major headache. Although his teams could create frequent releases, customers weren't upgrading. Engineering had to support multiple releases. Tom acknowledges that moving to major releases can help, but he still wants to use Agile development because the engineers are highly motivated, and teamwork is great, and direct customer feedback during sprint development has helped them build a great product.

Tom's developers have become disillusioned because he is unable to set schedules they can meet. The Agile teams don't create their own sprint objectives anymore. They must meet sprint commitments to support the feature schedules tracked by project managers. Developers barely have time to get the basic functionality working and don't have time anymore to create robust, high-quality software. Nobody seems to care as long as the features get checked off.

Product managers have been designated as product owners. However, they're never available when developers need to make design decisions. Developers just make assumptions because they must meet feature schedules. Developers don't get any direct feedback from customers anymore. The product managers feel they have enough experience to know what the customers need. They can just tell engineering what the customer needs.

Sprint reviews are tough. This is where developers are told everything that is wrong with what they've built. Many of the assumptions they had to make during development are ridiculed by product managers. They go away from the review with lots of changes, but the feature schedules stay the same. They're just going to have to let the defect backlog grow, hoping they have time before the release date to correct them. Developers don't include software for unlikely use cases and focus more on demonstrating the bare minimum to get them by the next sprint review. They hope to have time to account for these cases later. Unit testing is being bypassed to try to maintain the schedule.

Developers have started to leave the company. Tom's wife wants him to leave the company. He works long hours and comes home complaining about his day.

However, Tom has too much vested in the product and is loyal to the founders. He will hang on a bit longer hoping he can instigate some changes to make it more like the company he loved. He just doesn't know how to do that at this point.

1.1.5 Customer Perspective

AccuWiz was viewed as a breath of fresh air in the world of accounting software. They brought some novel ideas, especially around workflow to simplify the work for accountants. The decision to bring in AccuWiz was made by Adam, the head of his company's accounting department. He was willing to forego some of the bells and whistles of currently available accounting applications because AccuWiz made their routine work so efficient. They also got great support for small changes they wanted.

Their AccuWiz sales rep told them they were moving to annual release cycles. She explained that customers will benefit by getting more features with higher value, and customers will only have to upgrade once a year. Adam was okay with this because recent upgrades have surprised his staff with unexpected changes. They can now plan and do retraining prior to upgrading.

Adam is not happy with the way things have turned out. Only one of the ten features promised by their sales rep made the release. The release was three months late, and there were many quality issues. Adam noticed that the functionality they are seeing in this new release looks a lot like accounting packages they used in the past. He's wondering whether they just should have stuck with their old accounting software. The accounting staff is now angry about having to use AccuWiz, and their IT organization is complaining of the high level of support required.

1.1.6 Synopsis

AccuWiz represents the point in a company's life cycle where it transitions from entrepreneurship to bureaucracy. Scaling introduces organizational structure at the expense of communication and speed. Departmental goals become self-serving and often conflicting. For example, software development wants time to deliver with quality. Product management wants faster feature delivery and long-term roadmap commitments. Project management just wants to keep checking off features to meet the schedule, and sales wants every feature they believe can get them even one sale. The deeper the organization, the higher the organizational walls.

Of course, AccuWiz is a fictional company. It is made up from my early career experience in large companies and over ten years as a Construx consultant working with Fortune 500 companies. The pattern was observed over and over. There is intense pressure on engineering to commit more features than what they can deliver. Engineering questions the value of the endless requests for features, many

of which end up not being used by customers. Engineers can't meet schedules and become disillusioned and frustrated.

The pressure is not the fault of product managers. They are under pressure from the sales department. If you are a product manager in a large company, you may remember being told to force feature commitments on engineering under pressure from the sales department, often supported by the CEO – "don't come back without the right answer." It creates an antagonistic relationship between product management and engineering.

Let's also recognize that this situation is not the fault of the salespeople. Imagine that more than half of your income depends on making sales that depend on features being available on time. What would you do? Of course, you would use any leverage you have to push for features that could increase your income.

Another observation. If you switched the players around and ignore skill issues, what happens? Say Pete, the product manager, is moved to the sales department and Tom from development goes to product management. You will likely see them assume the same behaviors as their predecessors. You have likely observed this when people you work with have moved to jobs with different responsibilities. They start to behave differently. That means that there is something about how we define job responsibilities that impacts how people behave in organizations. We'll see that individual consequences drive behavior, and different roles have different consequences. We'll discuss consequences in detail in a later chapter on organizational behavior principles that will allow you to align role consequences with desired behaviors to change your organizational culture.

Does it have to be this way? Is there an approach that realizes the vision of Agile to deliver higher value faster with increased predictability in larger organizations? Can we create a culture of collaboration and teamwork between product management and engineering? We'll revisit AccuWiz after we describe an alternative that puts product management and development on the same side to achieve the Agile vision. The simplicity of the Investment approach will surprise you.

1.2 Summary

1. Financial reporting constraints drive the business need for multiyear predictability in terms of software delivery and development costs.
2. These constraints cause organizations to retain Waterfall planning practices that limit the business agility of Agile.
3. Organizational structures are typically based on the need for business predictability and divide organizations at the highest level with conflicting goals.

4. Engineers should be educated on real-world business constraints to create Agile development frameworks that meet the business need for predictability.
5. Knowledge and acceptance of real-world financial constraints usually explain the mysteries of business decision-making for engineers. "Follow the money." Accept it. It's not going to change.
6. Feature-level planning limits the value that Agile development can add by constraining solutions. It also leads to low-level feature commitments that are impossible to schedule with predictability.
7. The evolutionary requirements approach of Agile increases the likelihood that engineering will miss delivery and cost commitments, reinforcing a perception of poor productivity and lack of motivation.
8. The software industry needs to embrace the variability of software estimation to improve business predictability.
9. Replacement of Waterfall planning in Agile requires an approach that provides business predictability while retaining the tenets of the Agile development vision.

To make Agile successful in the eyes of the business, we need to understand why it is viewed as a failure in many organizations. That is the topic of the next chapter.