Turning Projects into Products (and Back Again)

Adam Light

adam@sotechadvisors.com

Abstract

Agile methods provide a product-centric model for software development. Projects, however, are deeply ingrained in corporate management systems and in the business models of service companies. Having established agile methods at the team or department level, many software development groups find further improvement hampered by conflicts with project-based policies, procedures, and practices.

How can you take full advantage of cross-functional teams and a product-centric implementation approach within a project-based planning process and in the face of policies and procedures that seem designed to prevent scope trade-offs? When software development is only one part of a larger work effort, how can you make a successful case for optimizing globally rather than locally? And with so many project, program, and product management roles involved, how can you maintain role clarity and a measure of stability even as you work to evolve those roles?

This paper presents a simple framework for successfully integrating project-based and product-centric approaches within a single, continuously improving workflow at the enterprise-scale. It describes an approach that explicitly defines the boundary between project and product-centric processes and maintains delineation as changes and improvements occur. If your organization depends on projects, you are experienced with Agile methods, and you have already tried the scrum of scrums, this paper may help you determine the next step to take.

Biography

Adam Light is Management Consultant and Principal at SoTech Advisors where he helps organizations apply lean and agile methods to unlock greater value from software development. Adam works with enterprise clients to adopt and scale agile methods, design and operate agile work systems, plan and deliver critical projects, and build the knowledge foundation necessary to sustain continuous improvement.

Adam has nearly twenty years of experience planning and executing complex software initiatives across a range of applications and industries. Prior to founding SoTech Advisors, Adam was Director of Planning and Program Management at TransUnion. Adam graduated with honors from Dartmouth College, holds a PhD. in Geography from the University of Oregon, and is a certified Project Management Professional.

1. Introduction

The Project Management Institute defines a project as "a temporary group activity designed to produce a unique product, service or result." (Project Management Institute, 2013) This definition emphasizes the unique aspect of projects and explicitly contrasts projects with routine work or operations. But many project elements may be viewed as routine work by the specialists who often provide services to deliver a project. In the development of software applications, for example, sizing and provisioning hardware may be accomplished by dedicated staff or outsourced to a provider of "cloud" computing services.

The adoption of Agile methods by software developers is causing many to re-think what they consider routine work. To be sure, knowledge work such as building software remains highly variable and still requires creativity, but stable cross-functional teams may be able to accommodate variation more readily, effectively, and reliably than projects can assemble a new group of people into a team capable of delivering a high-quality product. Many organizations and managers report shifting from a project-based approach that assembles groups of individuals to meet fixed scope and schedules to a product-centric (West and Wildeman, 2009) approach that focuses on maintaining stable self-organizing teams who deliver features and functions on a continuous basis.

But while product-centric development techniques are on the rise, projects remain everywhere on the landscape. Project-related concepts are often deeply embedded in organizational systems for selecting, funding, and monitoring work. A large enterprise, for example, may select and fund IT investments in the same regular annual budgeting cycle used to fund investments in facilities or factory equipment. And in relationships between companies, projects are the means by which service businesses usually transact with their customers. Contracts for outsourced software development, for example are often structured along project lines.

So why can't we simply combine project-based with product-centric approaches? Working together, shouldn't project managers and product developers be able to start out on a sure footing, deliver value rapidly, and stop when the time is right to move on to something else? Product and project-based approaches can co-exist within the same workflow, but the two approaches represent distinctly different strategies with fundamentally different assumptions and work best when the boundaries between them are clearly delineated.

This paper explains why "hybrid" approaches to delivering software that mix methods unintentionally often perform poorly, and presents an approach for successfully integrating project-based with product-centric business processes. After outlining the challenge inherent in combining the two approaches, I introduce the metaphor of air traffic control to explain the recommended approach and outline a framework for managing the flow of development and delivery initiatives through pipeline stages. I then present six steps to implementing a project delivery pipeline with a product-centric core.

2. The Challenge

At the center of the challenge lies a familiar paradox. Whether you are conceiving a ground-breaking new product or starting a book, beginning something new is often difficult, yet once you get going there is never enough time and you always wish you had gotten rolling sooner. Even if you have undertaken many similar initiatives before, the inertia of beginning frequently feels insurmountable. In what Donald Reinertsen (Smith and Reinertsen,1998) has famously called the "fuzzy front end" there are so many degrees of freedom that both the problem and the solution seem unbounded. A range of stakeholders and stakeholder groups with varying interests and perspectives often multiply the difficulties of finding a starting point. Product-centric and project-based methods approach this challenge differently.

Projects attempt to solve the problem of getting started by forging agreement on scope, schedule, and budget - the so-called "iron-triangle" of project management. If we can just agree on something (anything)

about what we want, when we need it, or how much it should cost, then – the argument goes – we will gain some kind of toe-hold and the basis for moving forward. Constraining design options may become a goal in the name of progress. Product-centric approaches, by contrast, acknowledge that starting is very hard and try to avoid the problem altogether by stopping as infrequently as possible, striving to operate a "design factory," through which product ideas flow.

Combining product-centric and project-based methods proves difficult because, in theory and in practice, the two approaches make different assumptions when accomplishing the same activities. Often those assumptions are not clearly stated or even consciously acknowledged by those who hold them. Friction emerges when people try to work together without clearly stating their opposing assumptions. Under such circumstances, unprincipled compromise does not necessarily make processes work better. Duplication of effort often results. In the worst case, incompatible approaches may result in different segments of the development workstream actively working against one another as factions form and one group strives toward the ideal of flexible scope while another strives toward the ideal of fixed scope.

The ability to increase value by varying scope lies at the heart of product-centric development methods. Well-crafted software built using modern languages and techniques remains malleable long into the development process. And the development process is as much a matter of learning and knowledge creation as a matter of construction. By fixing scope too early, organizations relinquish their main lever for value creation, while assuming that scope is going to change helps foster discipline on the part of developers by creating an incentive to emphasize flexibility and re-use over speculative feature development.

Variable scope forms the foundation of delivering business value with Agile methods. When flexible software is released frequently, along with mechanisms for rapid feedback to the people who are in control of the product scope so they can make trade-offs, the premise is that the best results will emerge. The evolution of technology makes this increasingly possible. In a world of connected devices, and with the decreasing overhead of releasing, software release cycles have shrunk and users have gradually learned to tolerate, expect, and even welcome regular software updates.

Between organizations that agree to share the work of software creation, however, it is not always possible or desirable to contract for a well-defined product from the outset. The value of supplier expertise often arises from the ability to frame problems and define the product, creating a chicken and egg situation. Counterparties frequently need to commit to working together before the product becomes well-defined so they craft agreements. Whether explicitly or implicitly, the "iron triangle" of project management is at the center of these agreements, often embedded in legal contracts, and this binds both sides to a project-based relationship. Scope that has been fixed early in the development process may prevent effective trade-offs that would later increase value.

In organizations of all sizes, established management systems assume the existence of projects. Budgeting processes often allocate funding by projects. Management controls release funds according to project milestones, staffing processes assign people to projects, and performance incentive plans refer to project outcomes. Companies that choose to capitalize software development base their accounting on guidelines that emphasize activities as much as the actual product of those activities. In many organizations, the existence of projects will be a fact for some time to come.

Formalized project management concepts run through many contracts and much has been written on the topic of creating Agile contracts (Arbogast, Larman, and Vodde, 2012). This article does not address Agile contracts directly, but if the proposed approach for reconciling project-based and product-centric processes within a single workflow can lead to more unified understanding of the development process, then contracts based on that understanding should better serve all parties.

While product-centric methods work better for some portions of the system delivery workstream, project-based processes may make sense for the other parts. And it may prove easier to embrace product-centric methods in stages. Organizations that can effectively integrate project and product-centric methods are able to start where they are and continuously improve from that point. The boundary between product and

project-based workflows can then be managed based on changing conditions and as a function of continuous improvement.

3. Air Traffic Control for Projects

For most people, the phrase “air traffic control” brings to mind an image of the people with binoculars who staff the tall towers standing above every major airport. In reality, however, the modern air traffic control system is more complicated than most travelers understand (Freudenrich, 2001)

As a plane moves from the gate at its airport of origin to the gate at its destination, it passes through multiple domains of control whose overall structure resembles the “zone” defense used by basketball teams. Figure 1 shows the path of a flight through these multiple zones. Each zone is suited to a different purpose and the total air traffic control system employs a well-defined set of protocols to knit the multiple zones together. Among other things, these protocols include the use of well-defined travel corridors and communication standards for handing off responsibility from one control center to the next. Key information about each plane and about the airspace has been standardized; controllers use the standardized information to coordinate air traffic under changing conditions.

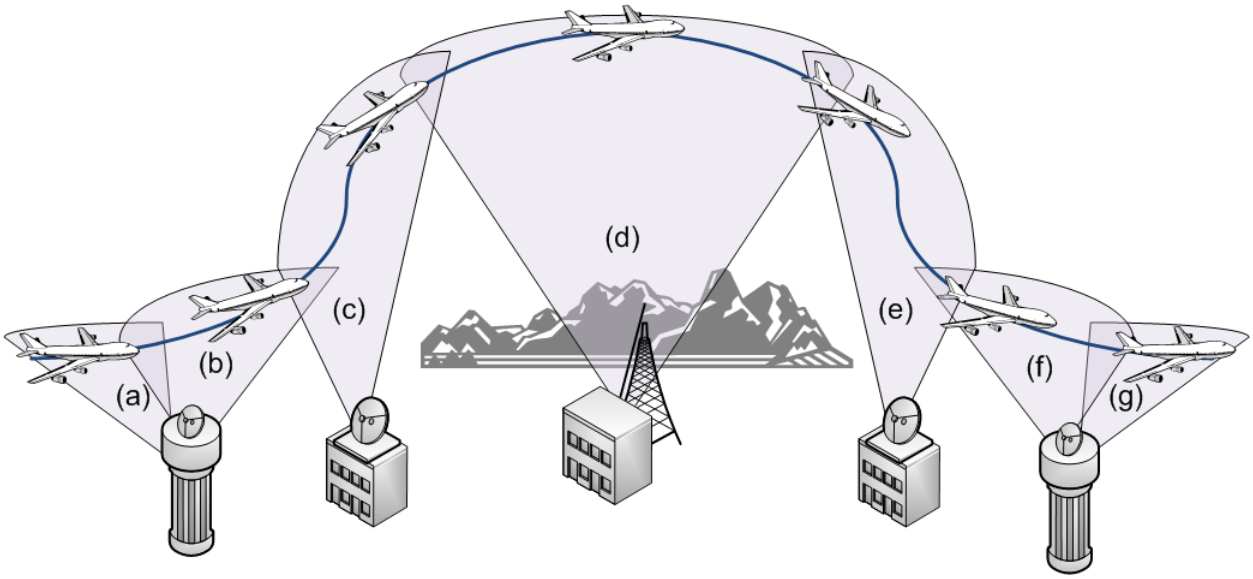

Figure 1. Flight path of an aircraft through air traffic control divisions.

The ground controller (a) is responsible primarily for the safe and orderly operation of traffic on the ground: getting planes from the gate to the runway as efficiently as possible. Once the ground controller completes his task, he hands off to the local controller (b) who then hands off to the departure controller (c) who hands off in turn to the center controller (d). Landing reverses the sequence (e, f, g).

The air traffic control system serves the safety and information needs of individual flights, but its more important role is to prevent collisions between flights and to minimize congestion. Maintaining safe and efficient airspace requires much more than providing a collection of uneventful flights. The complexities involved and the sheer size of the task (at peak times there may be 5,000 airliners in flight over the United States alone) make it impractical to simply dedicate a single controller to each flight over its entire journey. Such an approach would be both inefficient and inadequate for the task since it would simply magnify the challenges of coordinating so many flights. Its costs would be prohibitive and the system could probably not ensure safety or success across all the flights.

Within most large enterprises, and certainly between companies who contract with one another for services, sponsors and stakeholders tend to view themselves as customers in a service-provider relationship. They may escalate complaints to senior management when forced to make scope trade-offs or accept delays. But attending to each individual initiative or project cannot necessarily meet the needs of all the initiatives and all their customers. Modern enterprises are systems: large, differentiated, and complex. Individual projects frequently interact and conflict with one another, requiring trade-off decisions, and most projects must operate within constraints often shared with other projects.

Successfully delivering multiple concurrent and overlapping initiatives, then, requires a more sophisticated solution than simply assigning one (or more) project managers to each sponsored initiative. Regardless of how stakeholders view their separate projects, managing a collection of projects solely by dedicating individual project managers to serving individual projects and their sponsors over the entire lifetime of the project is no more adequate than dedicating individual air traffic controllers to serving individual flights. Such an approach is both inefficient and ineffective.

When a plane in flight experiences mechanical difficulties or encounters air turbulence, its pilots are capable of responding or adjusting course without direction from outside the plane. Like the flight crew of an airliner, self-organizing work teams, such as those defined by the popular Scrum framework (Schwaber, 2004), possess the necessary skills, experience, and resources to operate independently – both under routine circumstances and in the face of unpredicted events. The role of a project manager where work is organized in a product-centric model is to function like the center controller who monitors planes as they cruise across the country, the project manager can tune into information that the team doesn't regularly keep an eye on. The value added by the project manager in a product-centric model comes from maintaining a bird's-eye perspective and in monitoring the environment within which the team operates.

While a flight is on the ground, and during take-off and landing, by contrast, the ground controller, local controller, and departure controller monitor much more intensively and instruct the flight crew directly to keep the flight safely within established travel corridors. Like planes, projects are much more expensive in flight than at rest and they are much more likely to require guidance when taking off and landing. Project-based processes usually make the greatest sense in the beginning and ending stages of software delivery initiatives.

And flight crews are clearly not equipped to plan profitable routes, lease equipment, organize crew member schedules, manage fluctuating ticket prices, secure landing slots, handle luggage, or indeed to conduct any of thousands of other value-producing activities necessary for a flight to meet the needs of its passengers, crew, and airline investors. While these many disparate functions have been standardized in the operations of an airline, development work is highly variable and project managers usually find opportunity to add value by organizing and aligning elements surrounding the core development work.

In the past, demand for project management was highest during implementation, but with the advance of Agile methods this is no longer the case. To maintain the flow of development work, project managers should function more like air traffic controllers than pilots, deploying their resources in zones across the development cycle according to the needs of the organization and recognizing that continuous improvements in the flow of operations may cause their focus to shift over time.

4. Success Framework

While traditional approaches to project management gave the responsibility for assembling and aligning teams around the required work to the project manager, Agile methods form and maintain stable teams, devolving responsibility for alignment to teams and team members. What remains is the task of bringing the right work to the team at the right time.

The "ideal" Agile organization aligns stable cross-functional teams around specific products or product components. Where the work for any given team is characterized by a high degree of continuity over

time, the team can simply “pull” the next unit of work, and the organization functions like a factory with Agile teams as its work cells. In circumstances where self-organizing teams flourish, project management may play no role or may play a supporting role by providing specialized services to the team.

But where work is not characterized by a high degree of continuity, making appropriate work available to a specific Agile team at the right time becomes critical to success. Without a steady flow of available work, the Agile team cannot function smoothly and project management can play a central role in maintaining the flow of work.

The framework described below contains three basic elements: (1) the Agile “core”, (2) a pipeline for delivering work through this core, and (3) the ideal of continuous flow. With these three components in place, a stable, continuously improving system can emerge.

The Agile core

Giving responsibility to the development team typically works best when that team is stable and cross functional, and operates within an environment where it can control or influence things that enable it to carry out its responsibilities. Such conditions enable the team to experiment in order to find out what practices work most effectively.

Everything that follows in this article assumes the previous or concurrent implementation of Scrum or another approach to Agile work management. Organizations with any scale and longevity should also implement protocols for staffing, forming, and managing teams. And Scrum can only succeed to a limited degree without the tooling and knowledge to implement modern technical practices like continuous integration, test-driven development, and test automation. Empowered self-organizing teams form the backbone of Agile software organizations and coupling them with the right technical practices is the reliable way to improve productivity.

Making mistakes and taking accountability for not repeating them is an essential part of learning in cross-functional teams, and teams where a project manager regularly intervenes to prevent or curtail mistakes are less likely to exhibit effective self-organizing behavior. There is much more to forming and maintaining self-organizing teams than making day-to-day project management a team responsibility, but doing so is a prerequisite.

In the recommended framework, cross-functional teams make up the core of the pipeline through which work flows. Organizations that rely on projects but also want to achieve benefit from Agile methods should plan and prepare to explicitly re-deploy project management resources away from the Agile “core” as it forms.

Pipeline management

Managing a pipeline means tracking and actively controlling individual work items through a series of workflow stages in order to optimize flow and maximize the combined value of all work items delivered.

Pipeline management is often seen in sales functions where a typical business goal might be to grow the number of strategic accounts while maintaining or increasing revenue, or to maximize the total amount of sales in a particular time period. Pipeline management enables the sales organization to optimize globally rather than locally, and centralizing information in a sales force automation tool, such as Salesforce.com, helps managers maintain continuity when employee turnover occurs.

Making all the work visible in one place is a commonly cited benefit of pipeline management. Tracking and reporting on the pipeline may reveal patterns and help identify opportunities to simplify workflow. With a global view, managers (and other team members) can assess and respond to constraints in the work system rather than focusing efforts on expediting individual work items.

Pipeline management is also commonly applied to new product development which is modeled as a funnel whose purpose is to progressively develop ideas into a portfolio through active (rather than

passive) selection processes, but the extent and effectiveness of pipeline management often degrades as work flows from portfolio selection into implementation.

In response to the complexity of development work, many organizations form functional “silos.” In a product company, this may mean product management, engineering, and “go-to market” teams.” Within the engineering function, especially inside internal IT organizations, the silos may divide into functional units for quality assurance, business analysis, user experience design, operations and so on, and developers may also specialize in particular technical competencies such as integration or business intelligence.

Figure 2 shows an example of pipeline stages defined for an IT organization where project-based and product-centric processes coexist. Following project startup and once work has been transitioned to the Agile team and Product Owner, project management plays a supporting role. The PMO (project, program, or portfolio management organization) and business decision makers, actively manage the flow of work by tracking work in progress (WIP) and queue length in each pipeline stage to identify emerging bottlenecks and prevent overload in later stages. Once development completes, project and program managers re-engage in a leadership role to perform release management and launch activities.

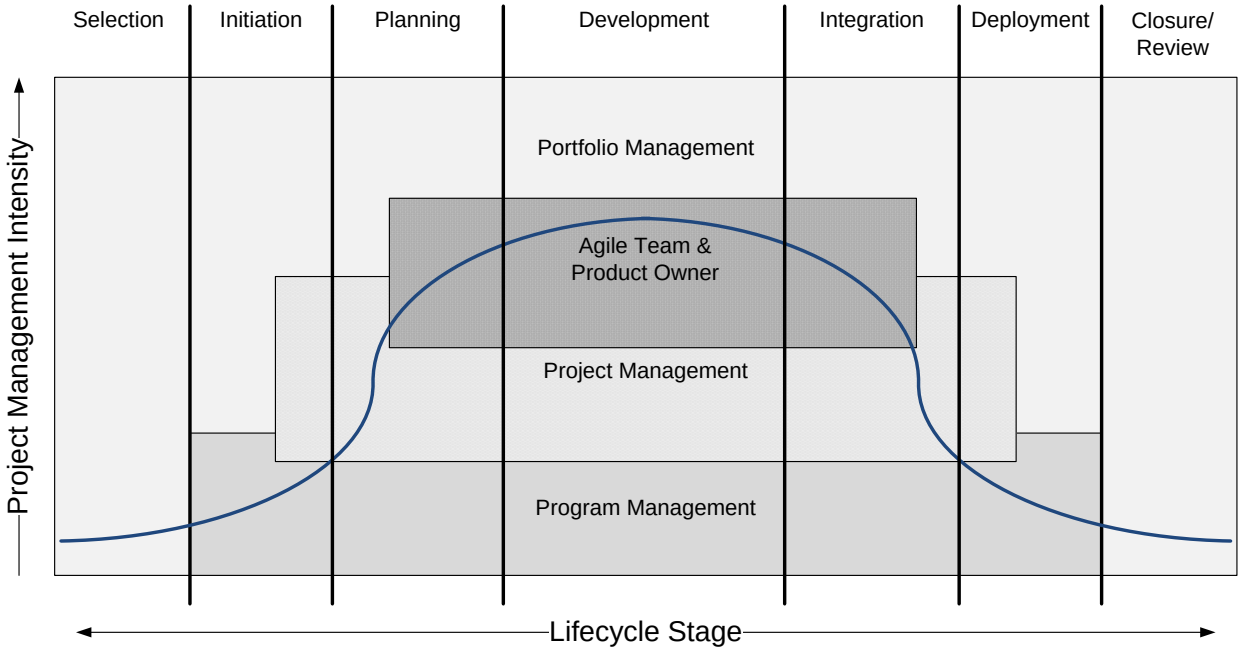

Figure 2 – Pipeline stages in an organization that integrates project-based and product-centric processes

Managing an individual project or initiative across functional silos is a familiar challenge to anyone that has ever worked in a development organization of any size, and project managers specialize in meeting this challenge. Related projects are often grouped into programs and PMO organizations work to “roll-up” projects and programs in order to provide visibility but as described above using the metaphor of air traffic control, focusing measurement and accountability at the level of individual initiatives does not optimize for the whole.

Managing a delivery pipeline implies assessing priorities and the performance of the overall portfolio at every stage and following policies and practices designed to maximize overall portfolio goals. Effective pipeline management can increase pipeline velocity – the rate at which individual work items move through the pipeline, and also improve throughput – the value or quantity of volume of items that move through the pipeline.

Pipeline stages, as illustrated above, may call to mind concepts made popular in software development under the heading of Kanban (Anderson, 2010). Kanban describes a specific approach to lean value stream improvement. The first step in Kanban is to make work visible and visualize workflow, which is exactly what the project pipeline does. The primary distinction is one of scale. While Kanban has most often been applied beginning with individual feature requests or even tasks, pipeline management looks at overall work in progress in the overall project portfolio and could therefore be seen as an initial step in the direction of Kanban at the enterprise scale.

The ideal of continuous flow

The term "lean" entered the vocabulary of business with the 1990 publication of *The Machine that Changed the World* (Womack, Jones, and Roos, 2007). Since the publication of "Machine," many authors have devoted themselves to studying the inner workings of the Toyota motor company. Among these is Mike Rother. In his 2010 book *Toyota Kata: Managing People for Improvement, Adaptiveness and Superior Results,* Rother describes the critical role of a guiding *process-based* ideal for continuous improvement at Toyota. This vision, which he describes as Toyota's "true north" for production can be summarized as "synchronized one-by-one (1x1) flow from A to Z at the lowest possible cost." (Rother, 2009, p. 45)

Value chain analysis and related lean techniques can help everyone see the big picture, but ongoing continuous improvement comes less from re-configuring the value chain and more from improvements at the detailed process level. Often, as easier improvement ideas are implemented, progress grows more difficult. Without a guiding vision like one piece flow, process-level improvements may oscillate between extremes, and improvements in different parts of the value chain may not be aligned.

The third element of our success framework, then, is the ideal of continuous flow that maximizes the total value of the portfolio delivered. Visionary statements can always be refined; the right way to state your initial vision initially is in a way that resonates within your organization.

5. Six steps for putting it into practice

Given an Agile core of cross-functional teams capable of operating in a product-centric manner, how should you implement a system for managing projects through this core? This section describes six steps for creating a work system that combines product-centric and project-based processes within a single workflow.

This article recommends first implementing a framework that clearly defines pipeline stages and the boundaries between project-based and product-centric processes, and then implementing protocols for making decisions about work flowing through the pipeline. Only after process boundaries are established and management aligned and educated should detailed roles and responsibilities be worked out for Project Managers, Agile teams and other actors in the different processes.

5.1 Build a coalition and communicate the vision for a pipeline model

The first step is educating decision makers on the concept of a project pipeline with an Agile core and, in the process, building support for subsequent steps. This step is extremely important, and should generally be the first step of any visionary change. This step corresponds to steps one to four in John Kotter's classic 8-step model (Kotter 2012.)

When implemented in concert with other Agile adoption or scaling efforts, the pipeline model is one component of the overall change. In situations where earlier Agile adoption efforts have not addressed governance and project management effectively and holistically, this change may stand on its own and coalition building may require overcoming disillusionment or misunderstanding about prior work.

5.2 Implement a master decision record

After obtaining buy-in and educating decision makers around the pipeline concept, the second step is establishing an effective framework for decision making.

Begin keeping a single record of important decisions about work and the organizational system that performs work. The structure of the decision record will evolve over time, but the most fundamental elements include the decision, the element of work or work system about which the decision was made (project, team, etc.), the decision makers who were present, and the information (presentation, report, etc.) that formed the basis for the decision.

Establishing a basic record of decision making is important for several reasons. First, it creates transparency that makes inconsistencies visible. The decision record provides a foundation for communicating effectively about decision status and outcome. Contradictory decisions also become immediately evident.

Secondly, ad hoc decision making represents a significant source of variability in many organizations. If two managers or teams independently make different decisions based on the same information, it may or may not be important to identify and reconcile the differences at the time decisions are made, or to standardize any particular decision type, but eventually, the organization will want to standardize certain types of decisions as guidelines, policies, or norms in order to ensure certain results and this will be impossible without understanding the variable processes that produced them. The decision record serves as a key basis for continuous improvement.

Last, and perhaps most importantly, culture change begins with behavior change. Few behaviors are as central to organizational culture as the way decisions are made and communicated. The simple act of making decisions explicit is a neutral action if done properly and in the context of making the pipeline visible. But asking who made a decision and what it was begins the process of rationalizing decision making.

5.3 Define explicit processes around making and communicating decisions

To avoid dilution of responsibility and a silo mentality that prevents flow, a single group or committee of decision makers should oversee the pipeline from one end to the other.

As with the decisions themselves, you cannot systematically improve the decision making process unless you make it explicit. While decisions that affect development may be made by different parts of the organization, some kind of single forum should be devised to review all decisions. A good starting point is the Operations Review format popularized in the Agile community by David Anderson (2003). Implementation can vary greatly but, consistent with the pipeline model approach to project management, agendas should focus on discrete segments of the pipeline. Referring to Figure 2, this might include separate segments for portfolio selection, initiation, planning, and development.

Complex modern organizations contain overlapping functions and pursue multiple aims simultaneously, requiring shared priorities and collective action. This typically means some kind of decision making committee. As with any team, there will be turnover, but you should keep decision-makers together throughout the pipeline

If structure and complexity did not already mandate some level of coordinated activity, providing consistent management to cross-functional teams requires management itself to become cross-functional. Whereas Scrum creates a container for self-organization and prescribes networks across which relationships form readily and reliably, the open-ended structure of committees contributes to their notoriously slow and uneven formation.

Why focus on something as seemingly unglamorous as committee protocols? Just like any other team, decision-makers need to learn to work as a team. Assuming realistically that some of the most important decisions in the organization will be made by committees of managers, and that those committees will take longer than teams to become proficient at decision making, it makes sense to (1) begin down the path to proficiency as soon as possible, and (2) aid the process by whatever means available.

Expressing processes as fractals is a common Agile concept. From an Agile standpoint, a committee is a team of decision makers who follow a more or less formal process to break down epics (projects) into releases and deliver them incrementally. Their work is accomplished through the direction they give to other teams and the decisions they make as a group.

5.4 Tie decision-making cadence to delivery cadence

Guided by a lean vision of one piece flow, value should be "pulled" by the customer and upstream processes should be pulled by downstream processes. The teams that form the pipeline are the work cells that produce value, so the cadence of decision making should be dictated by the iteration cycle of the teams.

If cadence is aligned, you can calculate decision cycle time from two important perspectives. First, if a team identifies a decision that needs to be made, what is the average or maximum time before the decision-makers will be able to convene and make that decision. Second, if decision-makers make a decision, how long before they are given feedback on that decision?

An important point to make here is the relationship between planning horizons and decision cycle time. Avoiding disruptions in flow requires planning at least as far ahead as the maximum decision cycle length and explicit acknowledgement that there is more waste from inaction than from action. That is to say, lack of preparedness for a decision should not prevent work in the short term.

5.5 Define project management roles explicitly

Projects are managed with varying degrees of formality in different organizations, and many authors use the term "project" in a relatively imprecise way to mean something like "the enterprise we're all engaged in to achieve a specific aim." Imprecise communication frequently obscures conflict between those who aspire to stabilize processes by improving project management and those who wish to institute and improve continuous processes. Once the pipeline is in place and decision making has been rationalized, project management roles can be defined precisely relative to specific pipeline stages.

Many times people start from the beginning of the business process to define roles, but it is better to proceed from areas of certainty to areas of uncertainty in the end-to-end flow. Also, it is better to start with the to-be process rather than the as-is process. Combining these two things implies that you should start with the product-centric part of the workflow and then define the handoffs into and out of it, agreeing to revisit the roles at a specified time in the future.

The most critical hand-off is from initial mode where scope may be fixed and schedule and budget vary to the condition where scope is varied to maximize value.

5.6 Pursue retrospective improvement toward the ideal

Just as an effective team is more than the sum of its parts, an effective organization is more than a collection of teams. Cross-functional teams lack defined positions and so they must form by working together and improving iteratively. When program and project managers are interacting with teams and decision makers to make good decisions, then all must be involved in effective retrospectives.

If the ideal is explicitly stated as product-centric flow, then at each stage you will try to chip away at the necessary wastes of project management; perhaps moving upstream to forge different relationships with sponsors in the client organization, or developing new policies for capitalization.

6. Making Room for Projects While Scaling Agile Delivery

Initially, Agile methods developed and evolved at the level of the development team. Building on the potential of dynamic, object-oriented programming languages, Agile methods demonstrated enormous potential to improve productivity and rapidly gained favor following the publication of the Agile manifesto in 2001.

A large part of the mechanism by which Agile methods grew rapidly in popularity was the marketing of Scrum through certification. Scrum became an accessible starting point for understanding Agile methods. Community support grew around Scrum as a sort of open standard. Scrum had many similarities to other Agile methods that were prevalent at the time of the Agile manifesto; for a variety of reasons it was simply Scrum that won out.

Since before the rise of Scrum, Agile practitioners around the world have been exploring what works for scaling Agile methods to the organization level. Fitting Scrum into the larger organizational context is a problem of great complexity. Scrum provides a blueprint for Agile work management at the team level; but deliberately avoids providing a similar answer to the question of what happens outside the team.

Given the practical and commercial success of Scrum, and the fact that there is no equivalent for Scrum at the enterprise level, it is perhaps inevitable that somebody would try to fill the gap. Two attempts to package and market frameworks for Agile methods at scale have recently gained attention: Scott Ambler's Disciplined Agile Delivery (DAD) and the Scaled Agile Framework (SAF) championed by Dean Leffingwell. While a thorough discussion of DAD, SAF, and related concepts lies beyond the scope of this paper, the rest of this section briefly compares the two and then explains how the recommendations above relate to these two topical reference points on scaling Agile methods.

There is a great deal of overlap between DAD and SAF in terms of the concepts and principles, and a great deal of intellectual weight combined behind them. This is good news for the field of software development because it indicates that the state of the practice is advancing rapidly. But with so many experienced, well-qualified thought leaders contending for mind-share and consulting business, there is bound to be debate and disagreement in any field

One long-standing controversy revolves around the certification business model. As mentioned, Scrum training and coaching have been extremely successful from a marketing standpoint. This helped spread effective practices and has enlarged the overall Agile pie greatly, but some parties have grabbed larger slices of the resulting pie than others, which inevitably excited some envy. Also, there is a legitimate point that the industry might be better served by a more diverse dessert table -- particularly as the center of gravity evolves beyond team-level Scrum. Proponents of the Scaled Agile Framework are, from a marketing standpoint, following in the footsteps of Scrum by using a training and certification model while the proponents of Disciplined Agile Delivery reject certification, branding their approach as "disciplined" while claiming to reject branded methods.

SAF focuses more explicitly on setting up a factory for development work. The Agile release train, which features prominently in SAF as the principal means of scaling development is roughly equivalent to what this article has described as the "Agile core" of the value chain. DAD acknowledges the release train as a valuable means for coordinating the work of multiple teams, but includes a variety of other building blocks and focuses on "tailoring" an approach based on multiple variables.

DAD and SAF both recommend product-centric approaches (as do Agile methods generally) and neither includes a specific role for project management. Both frameworks describe ways for accomplishing the traditional functions of project management through different roles. And their authors recommend that

current project managers transition into new roles or evolve their roles. In the case of SAF, the authors recommend that project managers move to a release manager role. DAD recommends that project management is a part-time role on most teams filled by one or more existing team members.

Taking SAF as the more concrete reference point, the approach described in this article can be seen as a specific way to implement effective processes for interfacing the product-centric Agile release train in software development with project-based work elsewhere in the organization.

7. Conclusion

In practice, project-based approaches tend to fix or control scope closely, while Agile methods are product-centric and attempt to vary scope in order to maximize value. The work of Agile adoption is in many ways the work of recognizing and coming to terms with this fundamental difference. This article's premise has been that, from a development standpoint, product-centric processes are optimal, but that practical reasons exist for project-based processes.

The nature and degree of project management support required outside software development will depend on the organization, but within the product-centric Agile core of the value stream, project management should typically operate more as a form of air traffic controller that serves the team and the overall work system. The hazard of a "blended" or "hybrid" approach is that it will be based on the lowest common denominator and that the benefits of each approach will be diluted. Instead we need an approach that maintains clear boundaries so that everybody can move continuously toward an ideal.

Since the two approaches make fundamentally different assumptions, maintaining them both within a smoothly operating workflow requires reconciling the differences explicitly. Where inconsistencies remain unaddressed, systemic misunderstanding may remain underground in the organization, erupting periodically in the guise of conflict over specific projects.

References

Anderson, David J. 2010. *Kanban: Successful Evolutionary Change for Your Technology Business.* Sequim, WA. Blue Hole Press.

Anderson, David J. 2003. *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results.* Upper Saddle River, NJ. Prentice Hall.

Arbogast, Tom, Craig Larman, and Bas Vodde. 2012. “Agile Contracts Primer,” http://www.agilecontracts.org/agile_contracts_primer.pdf (Accessed July 27, 2013)

Freudenrich, Craig. 2001. “How Air Traffic Control Works,” HowStuffWorks, http://science.howstuffworks.com/transport/flight/modern/air-traffic-control.htm (accessed July 27, 2013)

Kotter, John P. 2012. *Leading Change.* Cambridge, MA. Harvard Business Review Press.

Project Management Institute. 2013 “What is Project Management?” http://www.pmi.org/About-Us/About-Us-What-is-Project-Management.aspx (Accessed July 27, 2013)

Rother, Mike. 2010. *Toyota Kata: Managing People for Improvement, Adaptiveness, and Superior Results*. New York. McGraw Hill.

Schwaber, Ken. 2004. *Agile Project Management with Scrum*. Redmond, WA. Microsoft Press.

Smith, Preston G. and Reinertsen, Donald G. 1998. *Developing Products in Half the Time*, 2nd Edition, New York , John Wiley and Sons.

West, Dave and Roy C. Wildeman. 2009. “Product-Centric Development Is a Hot New Trend: A Distinctive Approach To Software Development That Delivers Exceptional Value,” Forrester Research, December 23, 2009.

Womack, James P., Daniel T. Jones, Daniel Roos. 2007. *The Machine that Changed the World.* New York, Free Press.