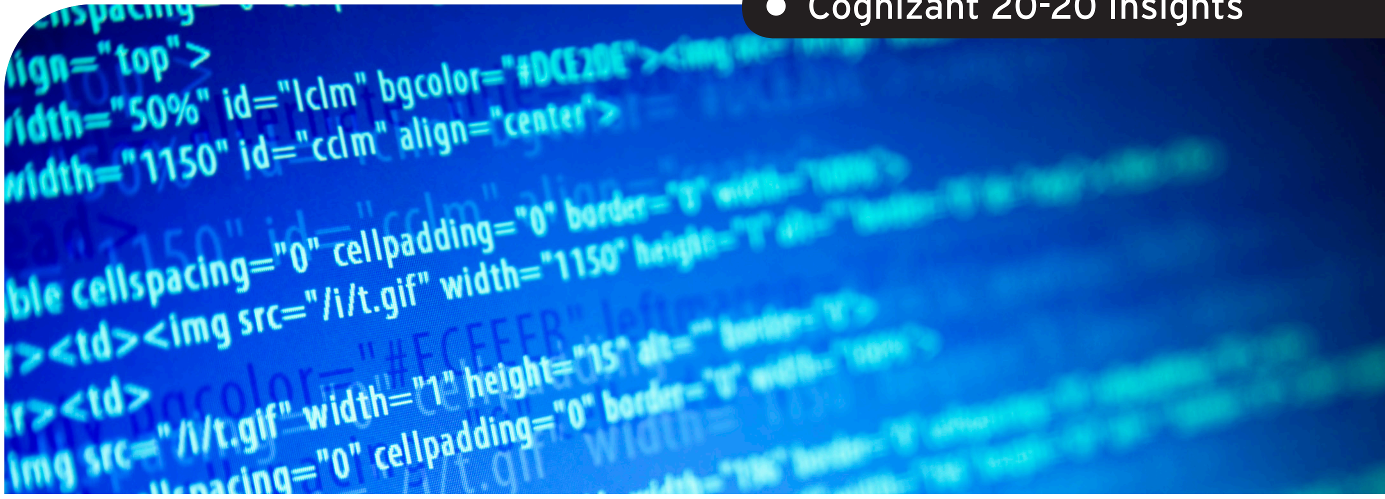

Kanban as a Tool in the Agile Toolbox

Executive Summary

The ability of a software development team to be agile has an attraction at an elemental level. Who would not want to "be agile"? On its face, it beckons. More to the reality of the industry, it was large-scale, long-running projects that badly missed the mark in terms of schedule, cost or scope that pointed out the need for new ways to organize and execute work. Slowly, responses emerged and a framework crystalized ten years ago when industry visionaries gathered to set a new course with a broadly-framed Agile Manifesto. The Agile movement was born out of frustration, not surface attraction. The quest for agility involves serious choices.

New practitioners are often reminded that "Agile is not a methodology." Approaches to task and team organization to better deliver business value through the software development process may involve a lightweight methodology as in Scrum or more simply be classified as a set of management policies as in Kanban. Either, for purposes of this paper, is considered a tool. Many combinations of methodology and policies have emerged over the last decade in the attempt to color in the picture broadly sketched by the Manifesto. Akin to constitutional case law guided by principles, real-world practices have been shaped by the variety of choices made by individuals and the organizations in which they serve. Tool selection and use are important choices.

Kanban is a more recent addition to the toolbox, with landmark use in the software engineering process at Microsoft starting in 2004 and maturing substantially since then. This paper examines the elements of Kanban as a tool, particularly in relation to Agile Scrum. In essence, the examination is an entry point deliberation, whether by itself or in combination with Scrum (i.e., Scrumban). The initial scope and complexity surrounding a Kanban implementation are indeed critical choices.

Tools for Agility

Overview

The Manifesto for Agile Software Development signed in February 2001 has proven to be a powerful statement of vision that has held together well over the decade and guided innovations in self-organizing teams. The Manifesto contains twelve principles[1] with four commonly-referenced attributes that reflect its essential spirit and vision:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Further, practitioners have noted two axioms that are central to agility efforts[2] and break from past practices of time-consuming, detailed front-end specifications and a lock-step, sequential implementation of all development efforts:

- **Axiom 1:** It is possible to divide the work into small value-adding increments that can be independently scheduled.
- **Axiom 2:** It is possible to develop any value-adding increment in a continuous flow from requirement to deployment.

With these guiderails in place for the path to agility, the selection and use of a methodology (e.g., Scrum) and a set of management practices (e.g., Kanban) are choices to be made in the context of the engagement and the culture of the organization.

Toolbox to Achieve Agile Principles

The literature is replete with admonitions from seasoned veterans that "tools are tools" — nothing more and nothing less. However, the policy decisions and choices surrounding the use of the tools is where the cookbook steps turn to craft. Organizations are well-advised to "measure twice and cut once" when selecting tools to achieve Agile goals. Anyone who has spent time in large-scale Agile projects or been enrolled when large organizations plunge headlong into Agile (i.e., "doing Agile") knows how diverting it can be to wrangle with unrealistic expectations and uneven levels of understanding surrounding the use of Agile tools. Agile coaches certainly have a role to play and many do it extremely well, but careful decisions around entry points into the Agile environments will pay dividends for those who have hired the coaches.

One primary reviewer cited a continuum along which the toolset lies ranging from very prescriptive (in terms of roles, steps and detailed instruction) to very lightweight. Scrum is on the right side of the continuum and Kanban serves as the right terminus. Methodologies like rational unified process (RUP), with its many roles, lie to the left.

In line with the Agile Manifesto assertion favoring "Individuals and interactions over processes and tools," the movement towards Scrum practices has become a standard entry point for Agile adoption. More recently, the simple set of powerful Kanban practices is enlisting a growing number of advocates recognizing that less is really more. This care and attention to the weight of prescribed activities centers on the degrees of freedom granted to self-organizing teams, which is a central consideration addressed by Manifesto principles:

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

A toolset that helps a development team or even an organization get to an established cadence of delivering high-quality software is the right fit. This obviously is a tailored fit based on the skill-set, motivations and experience level of the team; no cookbook recipes will work.

One of the key findings of this search for an appropriate entry point is that the adoption of the premise that "less is more" coupled with continuous improvement practices that are sincerely adopted by a self-organized team can build out to the level that meets the production goal.

Scrum

Basic Elements

There are many texts and blogs about Scrum, with detail for the curious, novice and experienced practitioners. A basic description works for this tool review:[3]

- Split the organization into small, cross-functional, self-organizing teams.
- Split the work into a list of small, concrete deliverables. Sort the list by priority and estimate the relative effort of each item.
- Split time into short fixed-length iterations (usually one to four weeks) with potentially shippable code demonstrated after each iteration.
- Optimize the release plan and update priorities in collaboration with the customer, based on insights gained by inspecting the release after each iteration.
- Optimize the process by having a retrospective after each iteration.

Change in Processes Required

Scrum is notable as a lightweight methodology. Its prescriptions include both team organization and practices.

- Organization: Must have a product owner, a Scrum master and a limited-size, cross-functional development team.
- Practices: Have a daily stand-up, deliver working product in fixed length iterations (i.e., Sprints), conduct Sprint planning meetings to establish a contract on what work will be delivered, and have retrospectives.

These mandates seem straightforward, but it is clear that some degree of organizational change is required at the onset. The reality is that even this limited set of prescriptions may be difficult to implement. The concept of an empowered product owner may be adopted in name only as decision-making in organizations of all sizes can be tricky; "final say" is often elusive. The change required for even this lightweight footprint requires broad-based if not universal understanding and full-fledged endorsement at all levels of the organization. The term "Scrum-but" is often heard in recounts of Agile adoption, signaling that organizational challenges and resistance to new practices are not easily conquered by those seeking to be Agile.

Kanban

Basic Elements

Lead industry practitioner David Anderson characterizes Kanban as a limited pull, work-in-progress system that is incremental and evolutionary.[4] Derived from the Toyota Production System (TPS) and introduced more recently into the software development world, a Kanban system is in essence a simple but powerful set of practices and policies. The term Kanban refers to a "signal card" that represents a unit of work that moves through the organization's work flow only when there is capacity to address the work at that step in the work-flow process (i.e., a "pull work system").

Almost every description of Kanban as a tool to manage and improve workflow starts with three basic elements:[5]

- Visualize the Workflow: A visual representation of the process lets you see exactly the state of the work activity (e.g., ready, in progress, done). A Kanban board is used which has a set of columns reflecting the steps of the work flow. With this tool, the work and the work flow is made visible to make activities and issues obvious.
- Work in Process: Kanban limits work in progress (WIP) through an explicit policy statement by the team to promote quality, focus and finishing (i.e., the team will accept no more than two active work assignments per team member at any one time). There is a limit to the number of things you can be working on and still do well.
- Measure and Improve Flow: Kanban promotes the measurement and tracking of workflow as a prompt for continuous improvement in order to continuously and predictably deliver value.

Each of these elements appears to be a simple construct, but veteran practitioners Mike Cottmeyer and Dennis Stevens report powerful implications and offer testimonials associated with the adoption of these elements even in the most basic form:[6]

- Visual control systems are valuable in changing behavior because they display status in an easy-to-see format. This ensures that everyone has a shared understanding of work status and process constraints. Transparency is a key to achieving organizational change.
- WIP limits encourage everyone to work as a team and prevent any one individual from getting too far ahead of anyone else.
- Kanban elevates awareness of constraints and forces the team to address them before they can bring additional work into the queue.
- New work is pulled into the system when there is capacity to handle it, rather than being pushed into the system based on demand.
- With Kanban, the focus of management and the team becomes the flow of work, not the utilization of the team.
- While Scrum has retrospectives at the end of each iteration to address these items, Kanban explicitly points out constraints in real time and encourages the team to address them as they arise.

As highlighted, the use of a basic Kanban board (manual and/or virtual for distributed teams) serves to bring everyone onto the same page, improve focus and prompt questions that lead to revision of policies and practices. When the team's experience grows, the Kanban board will change to expand and refine the workflow representation, thereby reflecting a more sophisticated level of adoption. Some examples of work-flow adjustments include:

- Insertion of additional queues between initially-defined columns to better articulate waiting states associated with bottlenecks and improvement opportunities.
- Use of a "priority next" queue to address the reality on the ground of development attention to some "must have" work items.

As well, policies associated with a work item, not just its positioning on the board, will also evolve as teams experience real-life situations that don't fit neatly on the Kanban board. That is, "classes of service" may need to be addressed by explicit policies agreed upon by the team.[7] For example, a class of service for a "live site bug" could have agreed upon rules such as:

- It is a top priority.
- It can break WIP limits.
- It can skip the design step.
- It can be released without product owner approval.
- It has an automatic same-day deadline.

No Change in Processes Required: Use as an Overlay

A particularly noteworthy attribute associated with Kanban is the potential to apply its use to an existing software development lifecycle or project management methodology. Kanban does not ask for a sweeping revolution of how people work; rather, it encourages gradual change. It's change that is understood and agreed upon by consensus among the workers and their collaborators.[8] As such, Kanban is a lean Agile system that can be used to enhance any software development lifecycle including Waterfall, which has implications for its immediate adoption in the application value management (AVM) arena.[9]

The literature review turned up many testimonials and endorsements of Kanban (such as the following) because the "up-front" organizational changes required are so limited:

- "Kanban is just about managing workflow. Initially, it doesn't replace anything the organization is doing. What it does do, however, is drive change. In Kanban you start with whatever process you have, visualize it, introduce WIP limits and then evolve from there."[10]
- "One of the great things about Kanban is that you apply it to your existing process. You are simply identifying ways to improve what you are already doing, so you don't have to start from scratch and you don't have to worry about "throwing the baby out with the bath water" – meaning that you won't lose the things you are already doing well. No sudden changes means there is minimal risk in applying Kanban as part of your improvement journey."[11]

Scrumban

Common Elements

When it is not used in a stand-alone mode or implemented as part of an AVM proposition, Kanban has been most closely associated with Scrum in practice. An overview of the similarities and differences[12] can suggest to teams seeking to improve their agility how the combination might work to their benefit.

Similarities reside in the fact that both:

- Are lean and Agile.
- Use pull scheduling.
- Limit WIP.
- Use transparency to drive process improvement.
- Focus on delivering releasable software early and often.
- Are based on self-organizing teams.
- Require breaking the work into pieces.
- Are continuously optimizing the release plan based on empirical data (velocity/lead time).

Why the Combination?

Explicitly limiting WIP is a key difference between the Scrum task board and the Kanban Board[13] (as highlighted above). Time-boxing, by its nature, sets a bound on how much WIP there can be, but it can still allow much more than would be desirable.[14] One common challenge in Scrum is late delivery within the Sprint. Late delivery introduces risk, tends to destabilize velocity and results in delayed value delivery to the customer.[15]

Author Corey Ladas suggests a number of intersection points between Scrum constructs and Kanban practices in his seminal writing, Scrumban:[16]

- "One simple technique that brings us much closer to our Kanban definition is to set a multitasking limit for individuals. You might have a simple principle like: prefer completing work to starting new work, or you might express that as a rule that says: try to work on only one

What is Different?

Scrum	Kanban
Time boxed iterations prescribed.	Time boxed iterations optional. Can have separate cadences for planning, release, and process improvement. Can be event driven instead of time boxed.
Team commits to a specific amount of work for this iteration.	Commitment optional.
Uses velocity as default metric for planning and process improvement.	Uses lead time as default metric for planning and process improvement.
Cross-functional teams prescribed.	Cross-functional teams optional. Specialist teams allowed.
Items must be broken down so they can be completed within one sprint.	No particular item size is prescribed.
Burn down chart prescribed.	No particular type of diagram is prescribed.
WIP limited indirectly (per Sprint).	WIP limited directly (per workflow state).
Estimation prescribed.	Estimation optional.
Cannot add items to ongoing iteration.	Can add new items whenever capacity is available.
A Sprint backlog is owned by one specific team.	A Kanban board may be shared by multiple teams or individuals.
Prescribes three roles (product owner/ Scrum master/team).	Doesn't prescribe any roles.
A Scrum board is reset between each Sprint.	A Kanban board is persistent.
Prescribes a prioritized product backlog.	Prioritization is optional.

Figure 1

item at a time, but if you are blocked, then you can work on a second item, but no more."

- "Another common technique is the late binding of tasks to owners. Some teams will pre-assign all of the known tasks during iteration planning. That's generally not a good idea because it artificially creates a critical path. Waiting until the "last responsible moment" to assign tasks to people maximizes knowledge and brings you closer to pull."
- "Another enhancement we can make to our previous board is to add a ready queue between the backlog and work-in-process. The ready queue contains items that are pending from the backlog, but have high priority. We still haven't bound any individual to these tasks, but as soon as somebody becomes available, they should take one of these tasks instead of picking something out of the general backlog. This enables us to decouple the process of assigning work from the process of prioritizing work, and it simplifies assignment. The ready queue also has a Kanban limit, and it should be a small limit, since its only purpose is to indicate which work item should be started next."

This is sophisticated usage of the tools without a doubt. And, it is hard not to conclude that the Scrumban nexus is best suited for those with a firm command of the principles, frameworks and management practices outlined above. The combination and orchestration suggested here is clearly the result of evolution, not top-down Agile mandates.

The end-state value suggests that the trip on the evolutionary path is worthwhile:

> "Transaction costs of negotiating scope and developing a schedule for each release was onerous" and "Effort estimate is turned back to productivity (analysis, coding, testing)."

Source: Anderson, David J. and Rick Garber, A Kanban System for Sustaining Engineering on Software Systems.

Assessment

Attribute	Category	Attribute
Stable and Certain	← Environment →	Turbulent + Unpredictable
Defensive Goal-Setting	← Strategy →	Proactive Learning Organization
Routine, Low-Discretion Roles	← Technology →	Complex, High Discretion Roles
Economic Work Orientation	← Culture →	Self-Actualizing Work Orientation
Mechanistic and Bureaucratic	← Structure →	Organic
Authoritarian	← Management →	Democratic

Source: Augustine, Sanjiv, Managing Agile Projects, Assess the Status Quo, p.109.
Figure 2

Kanban – Entry Point Considerations

As Agile adoption has grown, a number of implementations have met resistance, struggled mightily or been abandoned. Software development in matrixed, global businesses is a difficult task, to be sure. An understanding and appreciation of the level of organizational change associated with the introduction of Agility measures cannot be understated.

> "Change is a difficult business. People get attached to their habits and tribal affiliations. People resist changes that threaten their social status. Effective process change addresses human issues of fear and hope before interfering with workers' routines. Deming and the Lean thinkers have much to say about this. One thing you can do is to take an evolutionary approach and keep changes small and incremental whenever possible."
>
> *Source: Ladas, Corey, Scrumban – Essays on Kanban Systems for Lean Software Development.*

Given this reality, a set of considerations and advanced groundwork is advised with the introduction of Scrum:

With Kanban, the ability to start without wholesale process change has been noted. That makes its use attractive and accessible.

Lead practitioner David Anderson has outlined Foundational Principles (three), Core Properties (five) and Steps to Get Started (12). His articulation of Foundational Principles has extended the framework for Kanban's management principles outlined above.

Foundational Principles

- Start with what you do now.
- Agree to pursue incremental, evolutionary change.
- Respect the current process, roles, responsibilities and titles.

Core Properties

- Visualize the work flow.
- Limit WIP.
- Manage flow.
- Make process policies explicit.
- Improve collaboratively (using models and the scientific method).

Steps to Get Started

- Agree on a set of goals for introducing Kanban.
- Map the value stream.
- Define some point where you want to control input. Define what is upstream and who are the upstream stakeholders.
- Define some exit point beyond which you do not intend to control.
- Define a set of work items types based on the work requests that come from the upstream stakeholders.
- Analyze the demand for each work item type.
- Meet with the upstream and downstream stakeholders.
- Create a board/card wall to track the value stream you are controlling.

- Optionally, create an electronic system to track and report the same.
- Agree with the team to have a standup meeting in front of the board; invite upstream and downstream stakeholders (attendance not mandatory).
- Agree to have regular operations review meeting for retrospective analysis of the process; invite upstream and downstream stakeholders (attendance not mandatory).
- Educate the team on the new board, WIP limits and the pull system. Nothing else in their world should have changed.

Conclusion

Some level of Kanban implementation in pursuit of agility is advisable as the risks are low due to its very nature as an overlay and the ability to be implemented in multiple environments (e.g., Scrum, Waterfall, shared services and application maintenance work). The ability to use Kanban's core properties to guide team self-organization and as the basis for continuous improvement at a deeper and more refined level over time makes its use an attractive option when looking into the Agile Toolkit.

References

Anderson, David J. and Donald G. Reinertsen, Kanban: Successful Evolutionary Change for Your Technology Business.

Appelo, Jurgen, Management 3.0: Leading Agile Developers, Developing Agile Leaders.

Augustine, Sanjiv, Managing Agile Projects.

Azizyan, Gayane, Miganoush Katrin Magarian, and Mira Kajko-Matsson, Survey of Agile Tool Usage and Needs.

Hiranabe, Kenji, Kanban Applied to Software Development: from Agile to Lean.

Klipp, Paul, Getting started with Kanban.

Kniberg, Henrik and Mattias Skarin, Kanban and Scrum: Making the Most of Both.

Ladas, Corey, Scrumban - Essays on Kanban Systems for Lean Software Development.

NetObjectives, Lean Agile Project Management Course Material.

Rasmusson, Jonathon, The Agile Samurai: How Agile Masters Deliver Great Software (Pragmatic Programmers).

Footnotes

[1] http://agilemanifesto.org/principles.html

[2] Ladas, Corey, Scrumban: Essays on Kanban Systems for Lean Software Development (Kindle Locations 223-224), Modus Cooperandi Press. Kindle Edition.

[3] Kniberg and Skarin, Kanban and Scrum – Making the Most of Both.

[4] Anderson, David, The Principles of the Kanban Method.

[5] For example, Klipp, Paul, Getting Started with Kanban (Kindle Locations 23-25), Kindle Edition and www.kanban.com.

[6] Cottmeyer, Mike and Dennis Stevens, Kanban for Agile Teams.

[7] Klipp, Paul, Ibid.

[8] Anderson, Ibid.

9 "If you are starting from a phased, waterfall-type process and you have functionally-aligned teams, then you might start with a kanban process." Ladas, Corey, Scrumban: Essays on Kanban Systems for Lean Software Development (Kindle Locations 1067-1070), Modus Cooperandi Press, Kindle Edition, Further, the rhetorical question that follows suggests the direct applicability of Kanban over Scrum practices to the AVM environment: "Have you ever tried to talk about planning in two week iterations with a team that is responsible for production support? The idea that can't add anything to the sprint once it is started is going to fail – you'd probably abort every sprint." — Cottmeyer, Mike and Dennis Stevens, Kanban for Agile Teams.

10 Cottmeyer, Mike and Dennis Stevens, Ibid.

11 Klipp, Paul, Ibid.

12 Taken directly from: Bria, Mike, Comparing Kanban to Scrum, InfoQ for similarities and Kniberg, Henrik and Mattias Skarin,Kanban and Scrum: Making the Most of Both, p.50 for differences.

13 Cottmeyer, Mike and Dennis Stevens, Ibid.

14 Ladas, Ibid.

15 Cottmeyer, Ibid.

16 Ladas, Ibid.

About the Author

David E. Toombs is an Associate Director in Cognizant's Advanced Solutions Group (ASG) with over 20 years of experience with large-scale, strategic programs in a variety of industries. David has certifications from the Project Management Institute and the Stanford Advanced Project Management Program. He can be reached at David.Toombs@cognizant.com.

About Cognizant