

Kanban – An Alternative To The Agile Model Of Project Management

Rohith VISHWANATH IYER
ESC Rennes School of Business, France

ABSTRACT

Project Management has been used in various industries across all domains and it is necessary to follow some model to achieve the required results to involve all the stakeholders in the team, increase the efficiency of the team and also meet up the deadline set for the project.

Traditional project management are used in most of the corporations in the world to achieve the required result, but over the years many other systems were developed in place of traditional ones to increase the productivity and also experiment with the projects in order to achieve the desired result, one of them is AGILE and it is one of the successful systems in the world especially in service based companies and also many other lesser known systems like KANBAN is also available to manage the projects and also plan it with the best mechanisms to develop and make the project more productive and successful with also increased participation of the stakeholders in the team.

In this article, we will see how KANBAN is different from other systems and how it can be used to increase the productivity of the team by its application.

In the last section of the article, I am referring to my professional experience as a Scrum Master while working on a project in a multinational company using the Agile methodology.

Keywords: AGILE, KANBAN, Project, stakeholders

JEL classification: M1, M15

INTRODUCTION

Traditional project management tools were part of the industrial revolution from a long time mainly to maintain the required deadlines and due to the emergence of various other industries and application of the project management for all their projects, mismatch between the traditional project management techniques and the goals for the projects especially in IT Service Sector which involved some client based operations, a new system of project management mechanism was introduced called AGILE to facilitate the smooth movement of the tasks among the team members and delegation of the work and post-analysis of the team work to improve or include any new tasks for the team.

Traditionally AGILE has been in use in IT Sector for a very long time and it has been very successful also in managing the projects for a while but a new system in accordance with AGILE

called KANBAN is currently being experimented to more efficiently manage the tasks and also break up the required tasks with respect to the designated project.

Traditional Project Management Lifecycle Solution

Traditional project management structures focused on the paradigm of the project manager as controller

Traditional project management often was driven by formal reporting and hierarchical structures

Other features of it are given below:

- Centralization of Control
- Top-down planning
- Authoritarian environment
- Implied structure
- Limited/Restricted Access to the plan
- Local Access to information
- Limited Communications within team
- Separate projects
- Overly complex tools
- Rigidity of tools

A new solution called Project Management 2.0 was proposed to reduce the cons of the Traditional System and the differences between the two are given below:

Table no.1: Differences between traditional Project Management and Project Management 2.0

<i>Traditional Project Management</i>	<i>Project Management 2.0</i>
Centralization of control	Decentralization of control
Top-down planning	Bottom-up planning
Authoritarian environment	Collaborative environment
Implied structure	Emergent structures
Limited/Restricted Access to the plan	Organized/Unlimited Access to the plan
Local Access to information	Global/Live Access to information
Limited Communications within team	Unlimited Communications within team
Separate projects	Holistic approach
Overly complex tools	Easy to use tools
Rigidity of tools	Flexibility of tools

Source: <https://en.wikipedia.org/wiki/Kanban>

AGILE System

We saw from the traditional methods that the team did not have any co-ordination among them and project manager had the power to control each of them which leads to lot of confusion and deadline misses for the project. This called for a pro-team project management system which emphasizes on team players and non-hierarchical environment and also autonomy for the team members to execute the work delegated to them in their own way with minimal intervention of the project manager.

What Is Agile?

Agile methodology is an alternative to traditional project management, typically used in software development. It helps teams respond to unpredictability through incremental, iterative work cadences, known as sprints. Agile methodologies are an alternative to waterfall, or traditional sequential development.

What is Scrum?

Scrum is the most popular way of introducing Agility due to its simplicity and flexibility. Because of this popularity, many organizations claim to be “doing Scrum” but aren’t doing anything close to Scrum’s actual definition. Scrum emphasizes empirical feedback, team self-management, and striving to build properly tested product increments within short iterations. Doing Scrum as it’s actually defined usually comes into conflict with existing habits at established non-Agile organizations.

Scrum has only three roles: Product Owner, Team, and Scrum Master. These are described in detail by the Scrum Training Series. The responsibilities of the traditional project manager role are split up among these three Scrum roles. Scrum has five meetings: Backlog Grooming (aka Backlog Refinement), Sprint Planning, Daily Scrum (aka 15-minute standup), the Sprint Review Meeting, and the Sprint Retrospective Meeting.

Where Did Agile Come From?

In 1970, Dr. Winston Royce presented a paper entitled “Managing the Development of Large Software Systems,” which criticized sequential development. He asserted that software should not be developed like an automobile on an assembly line, in which each piece is added in sequential phases. In such sequential phases, every phase of the project must be completed before the next phase can begin. Dr. Royce recommended against the phase based approach in which developers first gather all of a project’s requirements, then complete all of its architecture and design, then write all of the code, and so on. Royce specifically objected to this approach due to the lack of communication between the specialized groups that complete each phase of work.

It’s easy to see how the “waterfall” methodology is far from optimized compared to agile methodology. First of all, it assumes that every requirement of the project can be identified before any design or coding occurs. Put another way, do you think you could tell a team of developers everything that needed to be in a piece of software before it was up and running? Or would it be easier to describe your vision to the team if you could react to functional software?

Many software developers have learned the answer to that question the hard way: At the end of a project, a team might have built the software it was asked to build, but, in the time it took to create, business realities have changed so dramatically that the product is irrelevant. In that scenario, a company has spent time and money to create software that no one wants. Couldn't it have been possible to ensure the end product would still be relevant before it was actually finished?

Why Agile?

Agile development methodology provides opportunities to assess the direction of a project throughout the development lifecycle. This is achieved through regular representation and monitoring of work, known as sprints or iterations, at the end of which teams must present a potentially shippable product increment. By focusing on the repetition of abbreviated work cycles as well as the functional product they yield, agile methodology is described as “iterative” and “incremental.” In waterfall, development teams only have one chance to get each aspect of a project right. In an agile paradigm, every aspect of development — requirements, design, etc. — is continually revisited throughout the lifecycle. When a team stops and re-evaluates the direction of a project every two weeks, there's always time to steer it in another direction.

The results of this “inspect, assess & adapt” approach to development greatly reduce both development costs and time to market. Because teams can develop software at the same time they're gathering requirements, the phenomenon known as “analysis paralysis” is less likely to impede a team from making progress. And because a team's work cycle is limited to two weeks, it gives stakeholders recurring opportunities to calibrate releases for success in the real world. Agile development methodology helps companies build the right product. Instead of committing to market a piece of software that hasn't even been written yet, agile empowers teams to continuously re plan their release to optimize its value throughout development, allowing them to be as competitive as possible in the marketplace. Development using an agile methodology preserves a product's critical market relevance and ensures a team's work doesn't wind up on a shelf, never released.

AGILE has its own advantages to steer the team towards success, but pushing in too many tasks at a time might be a risk and it is always necessary to understand from the team members about the roadblocks and also help each other perform the task in a more effective way, so KANBAN was introduced to reduce this as the teams work on only certain achievable tasks based on priority with deadlines defined for each and also understand the difficulty levels of the same and in case any of the task is completed before schedule, next task to be handled and if anything is getting stuck due to a roadblock, help the team negotiate with the blockage and close it before taking the next task.

KANBAN

Kanban is a new technique for managing a software development process in a highly efficient way. Kanban underpins Toyota's "just-in-time" (JIT) production system. Although producing

software is a creative activity and therefore different to mass-producing cars, the underlying mechanism for managing the production line can still be applied.

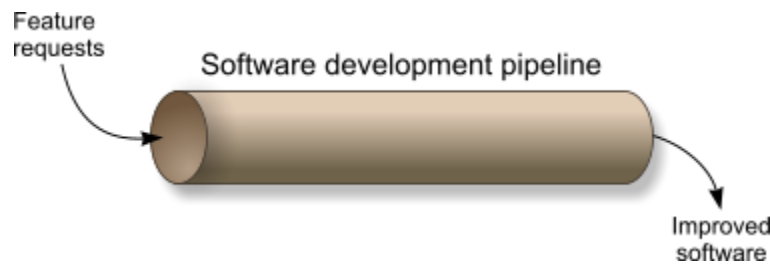


Figure no.1: Software development process description
Source: <http://kanbanblog.com/explained/>

In the representation above, the software development process is thought of as a pipeline with feature requests entering one end and improved software emerging from the other end.

Inside the pipeline, there will be some kind of process which could range from an informal ad hoc process to a highly formal phased process. It is assumed a simple phased process of: (1) *analyze* the requirements, (2) *develop* the code, and (3) *test* it works.

The Effect of Bottlenecks

A bottleneck in a pipeline restricts flow. The throughput of the pipeline as a whole is limited to the throughput of the bottleneck.

Using the development pipeline as an example: if the testers are only able to test 5 features per week whereas the developers and analysts have the capacity to produce 10 features per week, the throughput of the pipeline as a whole will only be 5 features per week because the testers are acting as a bottleneck.

If the analysts and developers *aren't* aware that the testers are the bottleneck, then a backlog of work will begin to pile up in front of the testers.

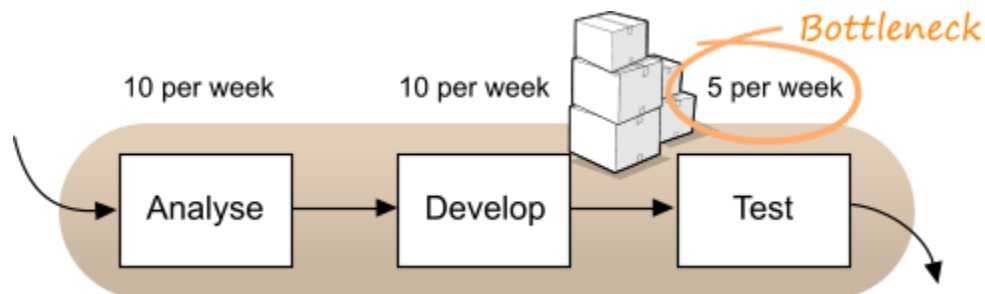


Figure no.2: Development pipeline example
Source: <http://kanbanblog.com/explained/>

The effect is that lead times go up. And, like warehouse stock, work sitting in the pipeline ties up investment, creates distance from the market, and drops in value as time goes by. Inevitably, quality suffers. To keep up, the testers start to cut corners. The resulting bugs released into production cause problems for the users and waste future pipeline capacity.

If, on the other hand, we knew where the bottleneck was, we could redeploy resources to help relieve it. For example, the analysts could help with testing and the developers could work on test automation.

But how do we know where the bottleneck is in any given process? And what happens when it moves?

Kanban reveals bottlenecks dynamically.

Kanban is incredibly simple, but at the same time incredibly powerful. A Kanban system consists of a big board on the wall with cards or sticky notes placed in columns with numbers at the top.

Limiting work-in-progress reveals the bottlenecks so you can address them.

The cards represent work items as they flow through the development process represented by the columns. The numbers at the top of each column are limits on the number of cards allowed in each column.

The limits are the critical difference between a Kanban board and any other visual storyboard. Limiting the amount of work-in-progress (WIP), at each step in the process, prevents overproduction and reveals bottlenecks dynamically so that you can address them before they get out of hand.

KANBAN with AGILE – SCRUMBAN

We can combine KANBAN with SCRUM and achieve better results by introducing the KANBAN board for AGILE and listing down the tasks as per priority with the task owner and moving the required tasks from one column to another without any bottlenecks created in between and also analyzing the task as per sprint in the regular sprint rules.

This has many advantages:

- Reduced backlogs of the tasks
- Regular sprint reviews with excellent time availability for other tasks in priority
- Task Owners are given enough autonomy to decide the severity of the task and prioritize them as per their requirement

- Better participation of the team in managing the project and setting, reaching the goals on time.
- Reduced participation of the project manager in the running of the team

My Experience as a Scrum Master

I was working in an MNC and my project was working as per agile methodology. I was Scrum Master of the team for 8 months and I was focused on making my team more productive as per requirements of the project and my tasks.

My key responsibilities:

I was involved in taking the availability of the team members for that sprint.

I was also involved in the delegation of the work as per availability of the team and allotting to the concerned task owner as per background and caliber.

It is always necessary for the scrum master to understand each team member's positives and negatives to delegate the exact work to that person to ensure that project is meeting deadlines and there is 100% employee skill usage for the project.

I was in charge with the prioritization of the tasks and putting it in the right order and also ensure they are finished in same order.

I conducted the Retrospective session to understand the good and bad part of the sprint and also review the sprint.

I was also collecting few important updates in the market related to our project that could be incorporated in the future sprints and also ensure continuous learning in the team with the project they are working on.

We as a team shifted to Scrumban and we on a demo basis, moved some of the tasks in the Kanban board and we started of this evaluation by putting the high time consuming tasks on the Kanban board and also ensured that in the daily scrums, we discussed the same as a highlight point to see what was the individual learning from this exercise.

I, as a Scrum Master, learnt a lot from this exercise and we as a team were successful in completing some high priority tasks but it takes a lot of time to switch from a conventional agile to Kanban system and it also depends on the mindset of the team to see that the flow is smooth.

CONCLUSION

Project Management is a big ocean and it is always necessary to draw the required water from it and also use it effectively – AGILE, KANBAN helps it to be done in a systematic way and it

also depends on various factors such as type of application, team size and complexity of the tasks designed for the project and always stake holders are very necessary for a successful implementation of the project and every team member with a stipulated autonomy for him/her makes the project more easy with equal participation and adaptation of new market trends to make the flow more productive in nature for all. Combining 2 frameworks will definitely give a good feedback and constant learning is the key to success of the project.

REFERENCES

James, Michael (Software process mentor), “The Scrum Reference Card, and Other Articles”, retrieved from: <http://scrumreferencecard.com/>

“What is Kanban?”, retrieved from: <http://kanbanblog.com/explained/>

“Agile methodology – understanding agile methodology“, retrieved from: <http://agilemethodology.org>

“Kanban”, retrieved from: <https://en.wikipedia.org/wiki/Kanban>

“7 agile certifications to take your career to the next level”, retrieved from:

VIDEOS

Intro to Kanban in Under 5 Minutes (What is Kanban, Learn Kanban), <https://www.youtube.com/watch?v=R8dYLBjITUE>

How to Build a Kanban Board, <https://www.youtube.com/watch?v=N3BoLRVXoI0>

The Kanban System, <https://www.youtube.com/watch?v=DdYdIMe0Sl8>