WHITE PAPER

Kanban execution: Optimizing work-in-progress (WIP)

Towards achieving a shorter lead time and better flow rate

Abstract

This is the second of a three-part paper on Kanban . In the first paper "Assessing Kanban fitment in the fluid and fast-paced world of software development" we explained Kanban principles in simple terms to make it easier to understand Kanban applicability in software development projects along with the various facets of Scrum and Kanban that should be considered while selecting the method. This second paper details the various aspects of work-in-progress (WIP) in the various workflow stages and how a Kanban team can optimize it to build a predictable outcome with balanced flow.

Infosys be more

A deep dive into Kanban execution

To move forward in the Kanban adoption journey, it is critical to understand the visualization of workflow stages in the requirement / task execution pipeline and the work-in-progress (WIP) limit in-depth.

Workflow visualization

The first step in implementing Kanban in the software development life cycle (SDC) is to visualize and plot the stages that any work item passes through in the project. If the work item is the user story, the left most column of the workflow stage will hold it, while the remaining columns can depict the task breakdown required to complete the user story. To ensure work items move on a daily basis / frequently, work breakdown must be done while staying true to the agile spirit, i.e., effort span should not be more than a day or a maximum of two days' worth work.

The workflow can be depicted on any information radiator such as the Big Visual Chart (BVC), whiteboard, or using a specific agile life cycle management tool such as Leankit Kanban, Swift Kanban, and more. Often, Scrum Board is used synonymously as Kanban board. While Kanban board may look like a Scrum board with more workflow columns, the major difference between the two is the WIP limit assigned to each column. A Scrum board generally has three columns – Ready, In progress, and Done as the team commits to deliver all work 'pushed' / selected in time boxed iterations, hence, tracking progress in more detail is not considered of much value.

Setting up a WIP

Once the workflow is in place, you need to address the most difficult challenge that the team faces – deciding the WIP for each workflow stage as it has to be based on the team composition and internal constraints under which a project operates. We will dive deep on how to decide WIP with a practical example in the next section. However, it is important to understand that Kanban execution progress is based on the Pull system. As soon as the number of work items in a particular column falls below the defined WIP limit, a new work item can be pulled in from the preceding stage. A work item that cannot progress to the next stage due to any constraint is still counted in the WIP limit and will certainly, slow down the workflow. The flow of work can be a complete 'hold-up', if all the work items get 'blocked'. In such a situation, the team will be forced to work with a single-minded focus on eliminating the blocker. This, in turn, would reduce multitasking and increase collaboration / team interaction as the team continues to practice Kanban and work with a disciplined policy of not breaching the WIP limit. The team can define explicit policies to breach the WIP limit based on their historical execution patterns to ensure that the team does not sit idle and revisit breach instances. Here, the team and all stakeholders must work as per the mutually agreed WIP breach policy, else it is easy to pull tasks more than the WIP which eventually can lead to failure of Kanban.

Strive for the perfect WIP limit

The real test for a Kanban team is to decide the WIP for each workflow stage (columns) which is similar to the enigma that the scrum team faces while deciding on the velocity to commit in the first sprint. In a traditional SDC, mostly, the team plan to start parallel tasking, considering that more WIP work items speed up delivery. But the focus of Kanban is on reducing WIP so that a work item moves towards completion faster. This is not easy to comprehend.

Opting for Higher WIP is akin to inviting complacency

More often than not, teams are tempted to choose a higher number for WIP as this would allow them to pick up new work items even when the task they are working on is blocked. As they say, the show must go on! However, think again. A higher WIP can lead to scenarios where team members may not require interaction. This would create knowledge silos and anychange in the team would then result in knowledge loss, preventing the team from delivering the work item with quality faster.

In addition, a higher WIP may jam the traffic in one stage, for e.g., Test In Progress. This may be the result of a skewed team mix where testers are less compared to developers. Similarly, the time taken to complete development and testing are not proportional so the supply of completed work might most likely be slower than what a tester can turnaround. In such a case, work items will pile up in the Test Ready stage.

Let's summarize the gain and loss in this case.

Gains	Losses
Flexibility to take up new tasks	Reduced focus on getting tasks unblocked (as it is not blocking the flow)
Workflow progress may not stick	Lead time of blocked work items can vary significantly
Maps with traditional SDLC, i.e., multitasking ways	Cycle time predictability reduces
Allows for independent working of the team	Less collaboration or one team
Continuous momentum for the team	Focus on multiple WIP work items leads to reduced amount of completed work
	Consolidated quality would be impacted

Is opting for a lower WIP, effective Kanban?

When a team starts on Kanban, skepticism on the lower number of WIP is bound to happen as they relate it to lower productivity. Initial thoughts might go towards opting for a lower WIP, in short, making the WIP equal in number to the number of people in the team working on the stage or slightly higher. But would that lead to the most effective Kanban? Not likely. In an enterprise application, distributed agile teams work in different time zones and a work item can move to the 'wait' state, as against being a blocker. Though a lower WIP will help the team focus on completing a task and keep it moving to the last stage faster. However, know that it will also uncover the workflow dysfunction very fast. Therefore, stakeholders should be comfortable to change and address them quickly.

Let's summarize the gain and loss in this case.

Gains	Losses
Playing by Kanban principles yields better outcomes	Team may feel stifled under constraints
Workflow bottlenecks would be visible very fast	Stakeholders should be prepared for change and address blockers faster
Smooth, continuous flow of work items towards 'Done' stage leading to optimum lead time	If workflow blockers are not addressed in time, there would be a loss of productivity (more idle time)
Single-minded focus on getting unblocked as the workflow might go to standstill faster	Enforced (or increased) breach of WIP limit
Improvement in team interaction and collaboration to complete other tasks when waiting for task blocks workflow	
Team policy would be reviewed frequently	

To summarize, the Kanban team must balance and understand the relativity of higher and lower WIP along with gains and losses in the context of project execution. The team can decide on accepting work items based on throughput and after duly considering training / skilling contingency for the person who just joined the project. As the team goes over and understands the nature of tasks, efforts, inflow, and team constraints, they move from `forming', `norming, and `storming' to being a `performing' team.

Let us take you through the scenarios to sync the WIP

In the case below, a team composition of one Designer, three Developers, and one Tester is assumed.

Option 1: Set the WIP limit equal to number of team members working on the work stage at any given point in time.

- WIP limit would be 1, 3, 1 for Design, Development and Test stages
- This looks good when the teams are collocated and the work will quickly move from one work stage to the next, however, to get the cost advantage, the teams work in a distributed mode.
- A distributed team working on enterprise applications in non- / less overlapping time zones more often than not, faces a situation where they are awaiting answers to pending queries from theclient. In such cases, the team experiences process losses in virtual teams where communication gaps appear even with a modern VC, Lync, and Webextechnology in place. This leads to some delays.

In such a scenario, setting up a WIP directly equal to the team number can stifle the workflow and force WIP breach instances. Therefore, we recommend setting up WIP as per case 2.

Option 2: Set the WIP limit = 2x where x is the number of people marked for that work stage.

- The WIP limit would be 2, 6, 2 for Design, Development and Test stages
- This gives flexibility to pull in a new task when a member is blocked on a task
- Based on the complexity of work items and categorized lead time taken by an average work item, WIP can be further fine-tuned for each stage as the team progresses and learns from the execution

Case illustration

Let us look at the below day-wise illustration of case 2 to understand the nuances of Kanban and how it uncovers bottlenecks.

Time	Backlog	Design In Progress	Dev Ready	Dev In Progress	Test Ready	Test In prog-ress	Release Ready
WIP		2		6		2	
Day 1	3	2		0			0
Day 1 End	1	0	2	0	0	0	0
Day 2	3	2		2		0	0
Day 2 End	1	0	2	0	2	0	0
Day 3	3	2		2		2	0
Day 3 End	1	0	2	2		0	2
Day 4	2	2		4		2	1
Day 4 End	0	0	2	4		0	3
Day 5	4	2		6		0	3
Day 5 End	2	2		0	6	0	3 (0 when work item released)
Day 6	2	2		0	4	2	3
Day 6 End	2	0	2	0	4	0	5
Bottleneck clearly visible, Designer WIP needs to increase to have better supply towards Development stage hence team shuffled composition with 2 designers, 2 developers, 1 tester							
New WIP	5	4		4		2	
Day 7	4	4		0	4	2	5
Day 7 End	0	0	4	0	4	0	7
Day 8	2	2		4			7
Day 8 End	0	0	2	2	2	0	7 (0 when work item released)

Legend: Red– underutilized WIP stage, Green – 100% utilized, Blue – Wastage (waiting or idle state)

In a steady state, Kanban work must flow smoothly and instances of task holdups should be reduced. In the example above, the team is underutilizing their Development capacity (waiting at development) since Design is full to the brim. A few observations below sum up the Kanban execution outcome.

- Only on the fifth day, the team was able to align with WIP (100% utilization)
- The analysis of Kanban flow discovered the design WIP as the bottleneck
- The team alleviated the constraint by changing the team composition, i.e., moving a Developer to the design stage
- Post the shuffle, the design capacity / WIP increased with an additional capacity to 4
- From their learning, the team also took the proactive step of introducing a WIP on backlog as 5 (1 extra than design WIP to ensure the designer has a steady inflow of work)

The team can benefit from the application of theory of constraints (TOC) which advises teams to focus on process / task in chain that is causing the delay in overall processing, causing low throughput. Here, the team should identify the bottlenecks: what is causing delays, underutilized states, and wastage? Joint brainstorming should be carried out by the team to come up with solutions to raise the constraints causing low output. It may be due to time- taking design reviews, complex systems and a new team, a new team member taking additional time, system access not given in time, downtime during certain periods, too many activities that the team is engaged in, and more. All this can help the team arrive at what can be an effective solution to weed out waste. They must then focus on these solutions and improve the system output.

For instance, at the end of Day 6, there are four tickets to be tested and due to two WIP in the Test state, only two tickets can be taken up. This implies that two

tickets are in the inventory waiting for a low testing capacity (perhaps, because the number of testers is less, or the access to the testing environment is constrained to only two, etc.). This means that whatever the development team may expedite will not lead to an overall throughput increase as the testing team's lower WIP is causing delays. In this case, the team should be convinced to get over this low WIP state at TEST and try to elevate to maybe, WIP of four. This can be done by reading through the trend of hold ups in the `Test ready' state. If this is consistent, it clearly shows that the capacity of the testing team needs to be increased by adding new testers, cross testing by developers, etc.

- Identify the system's constraint(s), e.g., maybe the testing team is idle, knowledge is localized, lead time is high due to pending access in system, clarification awaited
- Decide how to exploit the system's constraint(s) (how to get the most out of the constraint), e.g., if the testing team is idle, involve them in sharing production support tickets. If knowledge is localized, the team needs to be cross-trained on tickets
- Subordinate everything else to that decision
- Elevate the system constraints. Continue to improve

In a nutshell, all the time spent by tasks in orange in the 'Ready' state is the wastage which is increasing the lead time. To improve lead time, the team needs to reduce the wait time in the 'Ready' state and/or tune WIP. Kanban is effectively practicing elimination of waste. All the blue numbers show bottlenecks or idle time for the work items that are leading to low throughput and increased lead time. Therefore, setting a WIP would make or break the Kanban execution. The team must always aim to optimize it to achieve **smaller lead time, higher flow rate.**

About the Authors

Vikram Abrol

Lead Consultant, Infosys

Vikram Abrol is an IT professional with 16 years of rich experience. He holds PMP, CSM certifications. He is also affiliated with the leadership program at IIM Calcutta. He has been a Process Consultant and Quality Manager with seven years' experience at client sites in the US. He has been an Agile Coach for a large retail e-commerce program and has guided various engagements on Agile transformation journey, Agile maturity assessment, and consulting.

Ketan Navinchandra Shah

Principal Consultant, Infosys

Ketan Shah is a Principal Consultant, Agile and holds CSM, CSPO from Scrum Alliance, and Six Sigma Green Belt certifications. He has 14 years of IT experience which includes more than six years in E2E Agile execution from inception to implementation for several large enterprise engagements in various roles such as Agile Coach, Scrum Master / Mentor, Delivery Manager, and Agile Consultant. He is an Agile advocate, spreading Agile awareness via interactive trainings, mailers, community forum discussions, Agile COE initiatives, etc.

For more information, contact askus@infosys.com

Infosys
be more