Kanban

Lecturer: Ngo Huy Bien
Software Engineering Department
Faculty of Information Technology
VNUHCM - University of Science
Ho Chi Minh City, Vietnam
nhbien@fit.hcmus.edu.vn

Objectives

- To apply *Kanban* to develop software.

Contents

References

1. Eric Brechner (2015). Agile Project Management with Kanban. Microsoft Press.

Problem [1]

- We currently spend significant time doing work that's *unrelated* to delivering value to our customers.
  - We attend an excessive number of meetings about *planning* and *process*.
  - Careless team members are rewarded for pure speed, encouraging them to create costly bugs and submit *incomplete work*.
  - Schedules slip as *requirements change* and work is reprioritized, which forces more meetings about planning and process and wastes the effort spent on abandoned work.
  - *Quality* goes unchecked for weeks or months, which builds up an extensive amount of rework.
  - *Problems* fester for weeks or months before they are noticed, analyzed, and corrected.

Populate Your Backlog (Easy Way)

- Populating your work backlog is trivial if your feature team (3–10 people) is *part of a larger effort* and your work backlog is determined by your internal partners or customers (including bugs and tickets if you run an operational team).
- In that case, simply *write a descriptive name* for each of the features, improvements, and work items you need to accomplish on its own note card, and then place those note cards in the Backlog column of your signboard. (You'll prioritize them into buckets in a later step.) You're done.

Populate Your Backlog (Hard Way)

- If your team gets to *determine your own backlog* of work, you have some product planning to do.
- Product improvements (demands on your team) come from two sources:
  - *Your customers*: usage, experiment, feedback, and support.
  - *Your business*: Your leadership likely has ideas about how they want to improve the business, even if you work for a nonprofit. Your team, and your internal and external partners, likely have ideas about great new features and technology they want to add and bugs they want to fix.

Establish Your Minimum Viable Product

- The number of work items making up the MVP is a very *small percentage* of the total backlog.
- Typically, MVP items are *basic functionality* that customers expect before they'd even try your product, plus just enough differentiation to determine whether your new release is desirable.
- You sort all your items into the following buckets:
  - *Must have* MVP, sometimes called "pri 0"
  - *Should have* Priority 1
  - *Like to have* Priority 2
  - *Nice ideas* Priority 3

Order Work, Including Technical Debt

- Place *all pri 0 bucket items* (the MVP) in the backlog area.
- Related items within the same bucket should stay together, ordered by their natural *sequence of execution*.
- Items that can be started immediately (no dependencies) should be *ordered first*, followed by items that can be done soon afterward, and so on.
- After you place all your pri 0 items, if your backlog area still has room, begin placing the pri 1 items using the same approach.
- Take any pri 0 or lower items that don't fit in the backlog, and leave them in piles next to your signboard.

Step 1: Capture Your Team's High-Level Routine

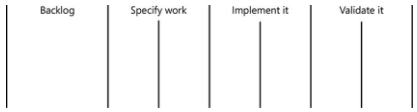
- Discuss the product with partners, teammates, and customers.
- Write and answer email, and engage in relevant social media.
- Write proposals.
- Find, evaluate, and fix bugs and operational issues (tickets).
- Track and design a response to feedback.
- Write major design documents.
- Produce improvements to products and infrastructure.
- Prepare for a major presentation.
- Roll out major changes.

Keep Your Routine Simple

- Include *only* the steps that your team does.
- If two sequential steps are usually done by *the same person*, combine those steps.
- When *in doubt*, use the steps that I use for my teams. These steps are fairly common: specify, implement, validate, and deliver.

Step 2: Redecorate Your Wall

- The first step, "Take an item from the backlog," is replaced by a holding area for pending work.
- The middle steps—specifying, implementing, and validating work—are divided into two columns each.
  - The left column of each step holds *active* note cards (items being specified, implemented, or validated).
  - The right column of each step holds note cards that have *completed* that step (items that have been specified, implemented, or validated).
- The last step, "Deliver it to customers or partners," isn't shown because that action is typically *handled in bulk* by a separate process.



Step 3: Set Limits On Chaos

- In a *traditional* Waterfall approach, chaos is limited by specifying all the work up front, enforcing a formal change-request procedure, and synchronizing work across teams at predetermined milestones.
- In *Scrum*, chaos is limited by planning time-boxed sprints, withholding plan changes until the next sprint, and synchronizing with the customer at the end of each sprint.
- In *Kanban*, chaos is limited by directly limiting the amount of work in progress (WIP)—literally, the number of note cards allowed at each step.

Determine WIP Limits — Fill in cells with yellow highlight

Step	Specify	Implement	Validate
A: Average rate per month per person	6	2	3
B: Slowest rate (minimum A column)		2	
C: Number of people assigned to step B		3	
D: Throughput of step B (B * C)		6	
E: People needed to match B's throughput (D / A)	1	3	2
F: WIP limits (E * 1½ rounded up)	2	5	3

Why a WIP limit of 1.5 per person in the team?

The Limits

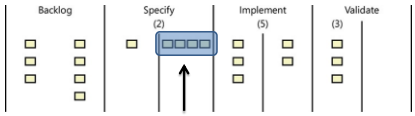
- The limits can be *adjusted* at any time to maximize team output and agility.
- The *WIP limit* applies to the total number of *active* and *done* cards for each step, except for the last step, which has a limited number of active cards but an unlimited number of done cards.

Step 4: Define Done

- *Specify done rule*
  - All items broken down into tasks that can be finished in less than a week each, and quick specs completed for each item.
- *Implement done rule*
  - Code is reviewed and unit tested, the static analysis is clean, the code is checked in, acceptance tests pass, and the customer-facing documentation is complete.
- *Validate done rule*
  - The work is deployed to production and tried by a significant subset of real customers. All issues found are resolved.

Step 5: Run Your Daily Standup

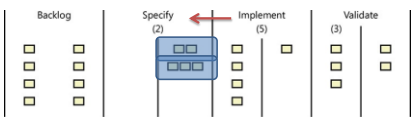
- The one required agenda item is asking whether any team members are *blocked* or otherwise *need assistance*, then assigning people (often the project manager) to resolve the issues.
- Experienced teams can complete the standup in *five minutes*.
- For an example of a standup, I'll pretend that the team *moves note cards* only during the standup.
- One item is being specified, and one item is done being specified (having been broken down into *four smaller items*, shown together on the same row).



less than a week each

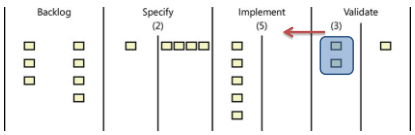
Troubleshooting (I)

- **Problem: Blocked because all items in an intermediate step are done.**
- In the example, the analyst was blocked because his step, Specify, had a WIP limit of two, he had completed two items, but the Implement step wasn't ready to take them yet.



Troubleshooting (II)

- **Problem: Blocked because prior step has no items done**
- The Validate step is ready for a new item (the last step's Done column doesn't count toward its WIP limit).
- However, the Implement step has no items that are done.



Troubleshooting (III)

- **Problem: Item blocked awaiting external input**
- Items often get blocked midstream awaiting external review, dependencies, questions, approvals, or other input from outside the team.
- To handle the issue, we add a Track column to the middle of the Implement step.

Troubleshooting (IV)

- **Problem: Step taking longer than usual for an item**
  - While different steps require different amounts of time and effort, each step should ideally take about the same amount of time for each item.
  - That's why the first step often breaks down large items to smaller items of similar size.
  - Team members should review what's happening
    - Perhaps the person assigned to the item is blocked or needs help.
    - Perhaps the item should be broken down further into smaller items.
    - Perhaps the person assigned to the item is expanding the scope of the item inappropriately.
    - Perhaps more *design work* is needed. **(See the next slide.)**
    - Perhaps there are some substantial *unresolved bugs*. **(See the next slide.)**

Troubleshooting (V)

- **Problem: Item needs design work**
  - Often this involves some user experience design, architectural design, and perhaps experimental design.
  - The Specify step for design work is breaking down the work into smaller tasks. The Implement step for those tasks is creating the design. The Validate step for those tasks is reviewing the design and getting it signed off. When it's done, you've got a completed design, which then adds several new items to the backlog.
  - Ideally, you'd use a separate signboard with steps specific to design work.

Troubleshooting (VI)

- **Problem: Bugs impacting team**
  - These tricky issues are managed like any other work item.
  - You create a note card, order it against the rest of the backlog, and slot it into its proper place.
  - These *complex issues* typically need to be specified, implemented, and verified like any other product work.

Troubleshooting (VII)

- **Problem: Important review, demo, or conference approaching**
  - The same way you account for design work—add a note card to the backlog.
  - Be flexible.
  - Define special done rules to suit the current work.

Troubleshooting (VIII)

- **Problem: New work, plan changes, and updated requirements**
  - Write note cards for new items.
  - Reorder the backlog to account for changes in priorities or to slot in new items.
  - Leave items already in work alone, except to edit them if their requirements have changed.
  - In the rare event that new work must commence immediately, you can move active cards to a *Track column* and free up space for the new items.

Troubleshooting (IX)

- **Problem: Item needs to be assigned to a busy team member**
  - Move the card to the Track column until the preferred team member is available.
- **Problem: Some team members like doing more than one item at a time**
  - Increase the WIP limit for this person's step by one or two (giving him 2–3 items at once). It's not ideal, and should be used only when the person's productivity is clearly negatively impacted by the lower WIP limit.

Troubleshooting (X)

- **Problem: Can't find time to improve tools and automation**
  - The solution is to put tool and automation improvements on note cards, just as you do for product improvements, and order them in your backlog accordingly.
- **Problem: New person joins the team**
  - WIP limits may need to be adjusted. Ideally, that person is assigned to the slowest step (increasing its throughput).

Estimate Features and Tasks

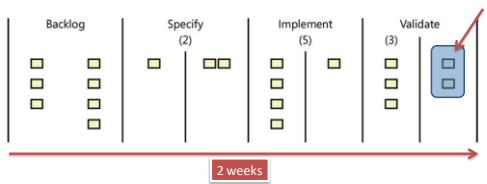
- **How soon will a work item be addressed and done?**
- **When will a significant product release be completed?**
- If your leadership, customers, or partners aren't asking these kinds of questions, there's no need to estimate features and tasks or track the expected completion date.
- Work items come in all different sizes, so it's hard to predict how long any particular item will take, let alone a whole backlog of items.
- *Breaks down* differently sized items into smaller, similarly sized items (typically taking one to five days each to complete). For clarity, I'll refer to these smaller items as *tasks*.

Planning Poker

- Each team member *privately* estimates the number of tasks needed to complete the work item in question.
- If the estimates *match*, you're done.
- If they *differ*, the high and low estimators explain themselves, the team members discuss their thinking, and then the process repeats until the estimates agree.
- *Repeat* the process for all the remaining work items.

Calculate Your *Task Completion Rate*

- Count *the number of tasks* that are done with their final step within any two-week to four-week period, and divide by the number of days.
- TCR = 2/10 = 0.2 (Note: 2 weeks = 10 days)



- **How soon will a work item be addressed and done?**
- **When will a significant product release be completed?**

When Will the MVP Be Released?

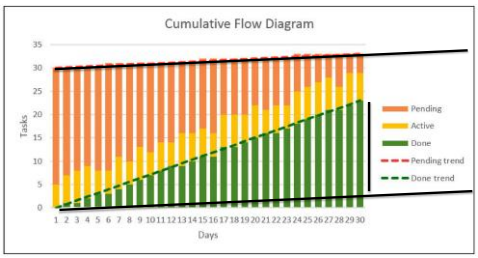
Question	Estimate (in days)
How soon will a work item be addressed and done?	(active tasks + estimated tasks for the work item and the work items ahead of it) / task completion rate
When will a significant product release be completed?	(active tasks + estimated tasks for MVP work items) / task completion rate

- The *product release estimate* is for completing the minimum viable product (MVP), not any pri 1 or 2 items.
- It's helpful for team members to compare the estimated number of tasks for a work item (written on the note card) with the actual number of tasks after that work item is specified.
- None of these estimates account for significant delays that result from *dependencies* or *changes* in plans.

When is the Completion Date?

- *Task completion rate (TCR)* You can track TCR as a moving average or monthly updated value (tasks completed per day). Doing so helps account for delays and changing team dynamics.
- *Current task estimate (CTE)* This is the total number of active tasks and estimated pending tasks (including tasks in a Track column). The CTE is essential for updated estimates because it represents remaining work.
- *Task add rate (TAR)* You subtract the total number of tasks (pending, active, and done) at the start of a month from the total number at the end and divide by the number of days. This gives you the tasks added per day (or tasks cut if the value is negative) and accounts for plan changes and feature bloat.
- With values for task completion rate, current task estimate, and task add rate in hand, you can calculate the number of days until your team will *complete* the current task set.
- It's CTE / (TCR – TAR). Why? – See the Cumulative Flow Diagram.

Cumulative Flow Diagram



From Scrum to Kanban

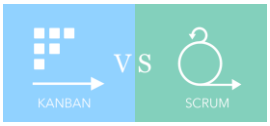
- In addition to displaying our backlog, *the board* will now
  - separate in-progress and completed work items for each step,
  - display our done rules for each step, and
  - have limits for how much work should be in progress for each step.
- The *work-in-progress (WIP) limits* and *done rules* will catch issues early and enable a continuous flow of work—one continuous sprint.
- Kanban *alleviates* the need for Sprint Planning, Sprint Review, and Sprint Retrospective events.

Is Kanban Scrum With a Board?

- Many Scrum Teams meet daily in front of *a board* that tracks their work progress.
- However, it's not Kanban unless you
  - list *your steps* on the board (like Specify, Implement, and Validate), and
  - each step has *a work-in-progress (WIP) limit*, a *Done column*, and
  - a *done rule* clearly marked on the board.
    - Our team decides on the done rules we'll follow. We know what causes issues. We know how to do our job well. We get to decide when a task should be considered done. Once we decide on the rules, we'll write them at the bottom of the signboard so that there's no confusion.

Scrum vs. Kanban

- With Scrum, we control the amount of unfinished work (work in progress) by *timeboxing our work periods (our sprints)*.
- However, careful time-boxing requires extra planning and doesn't always account well for the *variability of our work*.
- Kanban directly *limits the work in progress* of broken-down tasks. This eliminates the need for extra planning and makes the workflow smooth.



Scrum vs. Kanban

- https://kanbanize.com/blog/kanban-vs-scrum-infographic/
- Which one is better – Kanban or Scrum?
- To answer the question for yourself, you need to understand the difference between Scrum and Kanban.
- **Kanban vs Scrum – Roles**

- **Process Cadences**

Scrum vs. Kanban

- **Planning**

- **Commitment**

- As WIP limits prevent team members to work on multiple tasks at once, everybody commits to finishing what they have started before engaging in new work.
- In Scrum, the commitment for a Sprint is in the form of forecasting. When the team doesn't anticipate their capacity accurately or unexpected problems arise, either the sprint fails or personal heroics are required to finish everything on time.

Scrum vs. Kanban

- **Key Performance Indicators**

- Velocity is based on actual story points completed, which is typically an average of all previous sprints.
- Capacity is how much availability the team has for the sprint. This may vary based on team members being on vacation, ill, etc.
- Velocity charts are usually in the form of histograms showing the past performance of the Scrum team.
- Lead time is the period between a new task's appearance in your workflow and its final departure from the system.
- Cycle time begins at the moment when the new arrival enters the "in progress" stage and somebody is actually working on it.

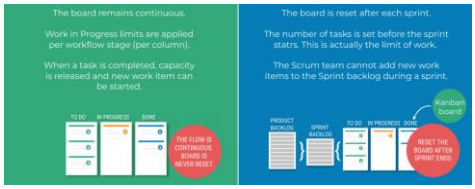
Scrum vs. Kanban

- **Meetings**

- In Kanban, meetings are optional.

Scrum vs. Kanban

- **Board**



Further Reading

- Richard Knaster, Dean Leffingwell (2018). Safe 4.5 Distilled - Applying the Scaled Agile Framework for Lean Enterprises. Addison-Wesley Professional.
  - SAFE.

Thank You & See You Again