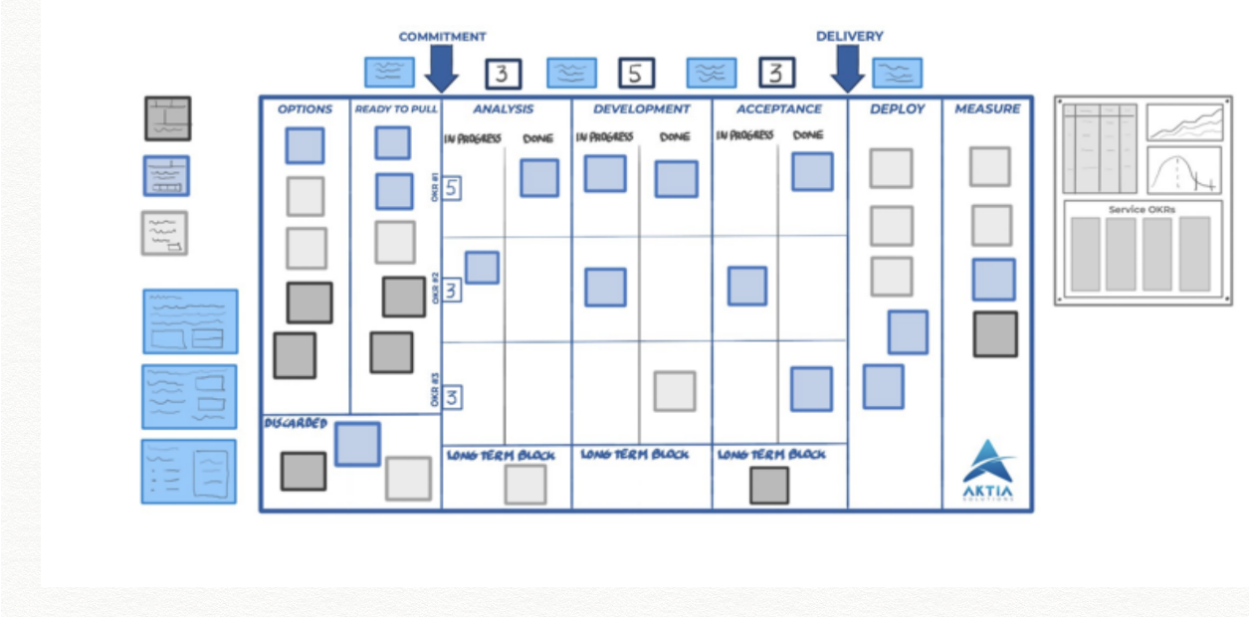
# Kanban essentials

Kanban is like the milkman. Parents didn't plan family milk consumption in advance for the whole year. Parents didn't give the milkman a schedule. They simply put the empty bottles on the front steps and the milkman replenished them. That is the essence of a **pull system**.

**Kanban Replenishment**
Kanban Systems can be replenished in two ways:
1. Constant time means they are replenished the same time each day or several times a day. This is similar to grocery store shelves being restocked each night.
2. Constant quantity is similar to the two-bin system. It may empty out at any time, and we refill it with the same quantity every time.

**Lean vs. Kanban**
The main idea of **Lean is elimination of wastes.** Anything that does not impact the functionality of the final software product positively is considered a waste in Lean.
Unlike Lean, Kanban is aimed not at eliminating wastes, but at **optimizing** the manufacturing process by regulating **the supply of raw material.** Visualization of workflow, limitation of work in progress, and iterative structure of projects are the main principles of Kanban.

Although it started in the software industry, Kanban is a method for defining, managing and improving all sort of services delivering knowledge work, such as HR, design or product management.




Kanban is based on a very simple idea, which in our experience is counterintuitive and difficult to grasp for many people: **Work in Progress (WIP) should be explicitly limited** and something new should be started only when an existing piece of work is pulled by a downstream process.

**There is a famous saying by Kanban acolytes which says, "Stop starting, and start finishing".**

Because WIP is limited in a Kanban system, **anything that becomes blocked for any reason tends to congest the system. If enough work items become blocked the whole process gets to a halt.** This has the effect of focusing the whole team and the wider organization on solving the problem, unblocking the item and restoring flow.

Reduction of WIP leads to all the Kanban benefits:

- Reduction of time-to-market
- Reduce overburden
- Reduce waste of over-production
- Productivity
- Predictability
- Customer satisfaction
- Enables continuous improvement

**Productivity improves by doing less things at a time but finishing more.** The focus in Kanban is on making items flow as fast as possible through the system.

The symptoms when Kanban might help:
• the future is uncertain
• priorities are often changing
• **replanning is common and frequent (even within the same sprint with the addition of urgent work or bug fixes)**
• high discard rate
• high cancellation rate of already started work
• there is a significant amount of delivered work which is ignored or made available to customers much later than the actual delivery date

**Do you have problems with the flow?**

- Are you constantly asked to **start new work before you have had a chance to finish old** work?
- Are you constantly asked to **expedite new work in addition to being expected to get all of your other current work done** according to original estimates and commitments?
- How many features do you **start but do not finish because they get cancelled** while you are working on them?
- When something that you are working on **gets blocked, do you simply put that blocked work aside** and start to work on something new?
- Do your **estimates give consideration to how many other items will be in progress** at the time you start work?
- Do you **ignore the order** in which you work on items currently in progress?
- Do you **constantly add new scope or acceptance criteria** to items in progress because it is easier to modify an existing feature rather than to open a new one
- When an item takes too l**ong to complete**, have you ever said or heard someone say “it is just bigger than we thought it was” and/or “it will get done when it gets done”?

If any of the questions above apply to your context you definitely have a problem with flow.

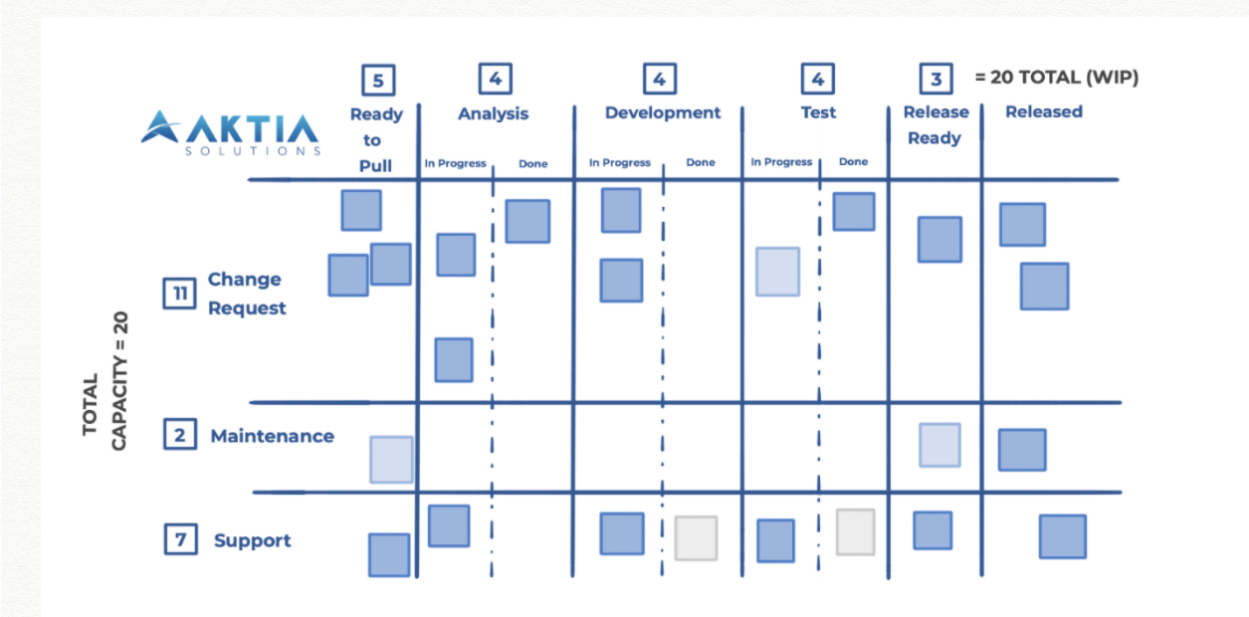
**Variability in the Flow of Work**
Variability in flow can occur for many reasons; for instance: **large** batches, by arrival of **unplanned work** in an unpredictable fashion, or by **blockages** which prevent work from being completed.

Unplanned work often comes with a *high class of service*. This results in planned work in progress being set aside in order to service the unplanned work. I am quite sure you can recall situations like *“the CEO asked me” or “this important stakeholder told me to do this urgently”.*

**Classes of service**

**By Work Item Type:** maintenance, development, bug, report, support …
**By Cost of Delay**: standard, expedite, fixed date, intangible
**By customer type:** *Customer A values quality; Customer B values time to market, Customer C values low cost*

**By project**
**OKRs**
**Go-to-market strategy**




**Basic Kanban Metrics**
*Lead Time, WIP and Throughput.*

- **Work In Progress** (the number of items that we are working on at any given time)
- **Cycle Time (Lead Time)** (how long it takes each of those items to get through our process from commitment to delivery)
- **Throughput** (how many of those items complete per unit of time). The unit of time that your team chooses for your Throughput measurement is completely up to you. For example, *User Stories / Week* or *Features / Month* or *Experiments / Day*.

**Little's Law**
avg. cycle time = avg. wip / throughput
Shouldn't be used for predictions.

**Flow Efficiency**
Flow efficiency % = work time / lean time *100%
*Flow efficiency* is the Kanban metric that measures the percentage of total lead time is spent actually adding value (or knowledge) versus waiting.

**Flow Debt**
We also refer to flow debt as **aging** work, which means that an item has been in progress for longer than the average cycle time. When an item is accumulating flow debt it is indeed artificially aging. As we said, this happens when work in progress is left aside to work on other items that entered the system later.
When aging is higher than lead time it is a clear sign that your system it not stable and it is not flowing properly. **If your Aging is much higher than your Lead Time, you have a problem! In a stable system percentile 85% of lead time and percentile 85% of aging should be similar.**

Ask yourself:

- Why did any of that work got into the system in the first place?
- What is the criteria for committing to deliver on something?
- Why aren't we working on the old tasks any more?
- What is preventing us from finishing old tasks?
- Why are we keeping things in progress and delaying them? Could we cancel those? Could we deliver them?
- Why are we adding more stuff if we still have a lot of stuff that has been there for weeks?

**Discard Rate**
Discard Rate is the Kanban metric that measures the number of options that are discarded, and thus eliminated from the upstream process, before commitment point.
**Effective organizations show high discard rates in all steps of the upstream process and teams's backlogs are not cluttered with hundreds of options.** On the other hand, companies with low discard rate are usually led by opinions, intuition and don't have a framework for developing options and deciding what to work on.

**Blockers**
Dependencies and blockers stop the normal flow of work and introduce a lot of variability in the system. Something that is evidenced with a long tail in Lead Time histogram.

**The Core Practices of Kanban**
1. Visualize
2. Limit work-in-progress
3. Manage flow
4. Make policies explicit
5. Implement feedback loops
6. Improve collaboratively, evolve experimentally (using models & the scientific method)

**Getting Started with Kanban**
The essence of starting with Kanban is to **change as little as possible** in your existing process.

The main target of change will be the **quantity of WIP** and the **interface with upstream and downstream** parts of your value stream.
You must work with your team to map the Value Stream as it exists

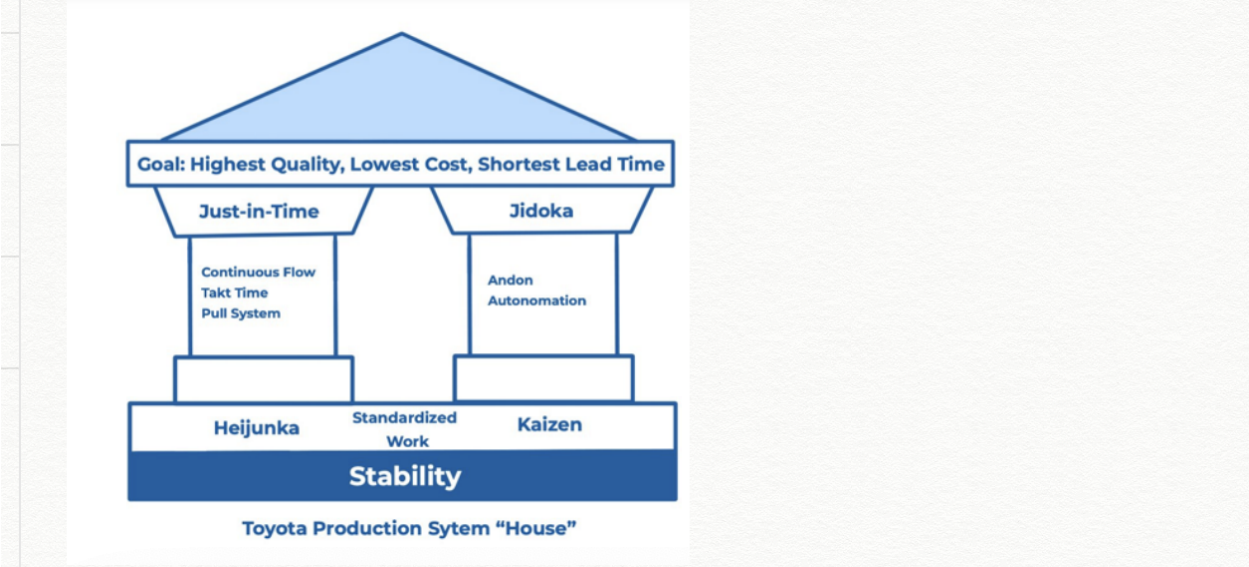
1. Agree on a set of **goals for introducing Kanban**
2. Map the **value stream**
3. **Define** some point where you want to control input (**Commitment Point**)
4. **Define** some exit point beyond which you do not intend to control (**Delivery Point**)
5. Define a set of work item types based on the **types of work** requests that come from stakeholders and customers
6. Analyze the *demand for each work* item type
7. Agree policies with upstream and downstream processes: agree on a WIP limit, agree on replenishment process & when to initiate it, agree on a delivery coordination mechanism, such as regular software release, introduce the concept of different classes of service, agree on a Lead Time target for each class of service and/or work item type,
8. Create a dashboard visualize and monitor the value stream
9. Agree with the team to have a standup meeting every day (Daily Kanban Meeting)
10. Agree to have a regular Team Performance Review meeting for a retrospective analysis of the process, the product and the customer.

**Lean**
Lean is about building and improving stable and predictable systems and processes to **deliver to our customers high-quality products on time** by engaging everyone in the organization.

We need to create an environment of **respect for people** and **continuous learning**. *It's all about people.*

People create the product and drive innovation, and with **leadership buy-in and accountability to ensure sustainment with this philosophy**, employees will be committed to the organization as they learn and grow personally and professionally

Everything we do nowadays like Agile, Scrum, Kanban, OKRs or Lean Startup has its roots in Lean.




The purpose of Lean organizations is to achieve the sustainably shortest lead time with best quality and value to people and society, high morale, safety and customer delight

To achieve the goal, Lean organizations rely on the two pillars of **Respect for People** and **Continuous Improvement**, and the practices and principles of product development and product manufacturing.

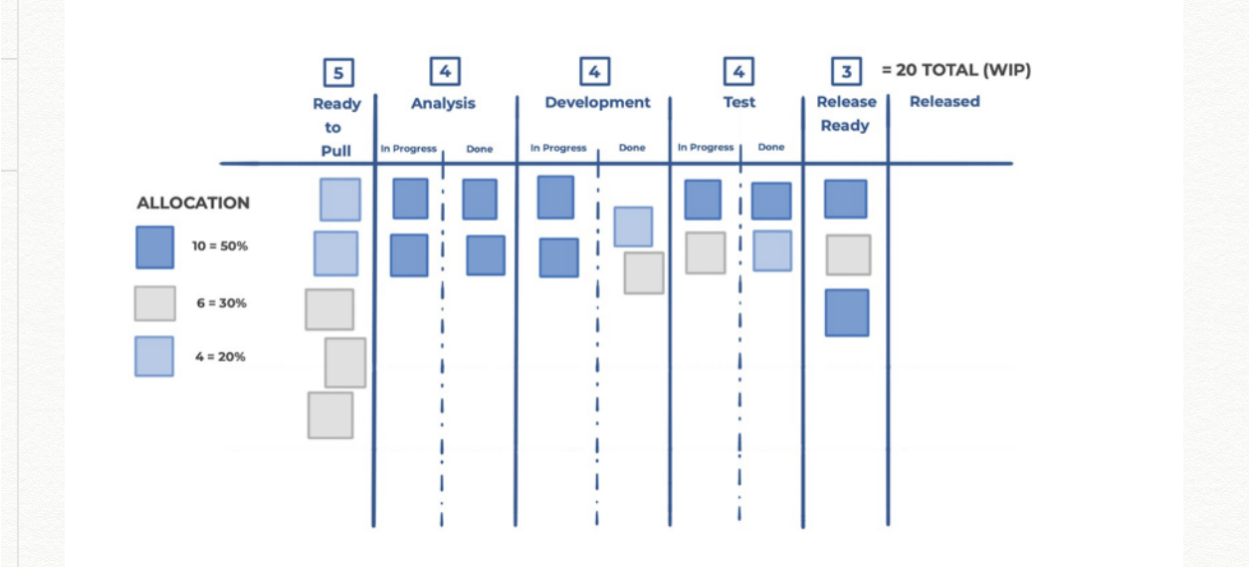
The six keys to improving product development flow are:
1. Utilization
2. Variability
3. Batch size
4. WIP Constraints
5. Pull
6. Cadence

Operating a product development process near full utilization is an **economic disaster**.

**Reducing Variability in Kanban**
Below you can see some techniques used in Kanban to reduce variability:
• Ready to Pull policies
• **Classes of Service**
• WIP Limits
• Visualization policies for blockers and dependencies
• Flow management policies

**Batch Size**
There are two factors that combine to cause many of the problems we associate with traditional software development, the size of work (batch size) and the separation of responsibilities (silos).
**The bigger the batch, the bigger the queue.**
Big batches also bring additional problems like low quality, longer feedback loops and hidden variability.

**Applying WIP Constraints**
- adopt local WIP limits, like we do in Kanban by limiting work-in-progress by column.
- limit WIP by class of service or work item type
-on top of those two mechanisms, we have to make WIP always visible
- analyze aging of work items and define proper way to manage those items that are accumulating flow debt
- discard options as soon as possible by implementing clear selection criteria
- prevent the uncontrolled expansion of work by defining policies that indicate when to finish a project. If you focus on outcomes you might decide to finish when the outcome (goal) is achieved.

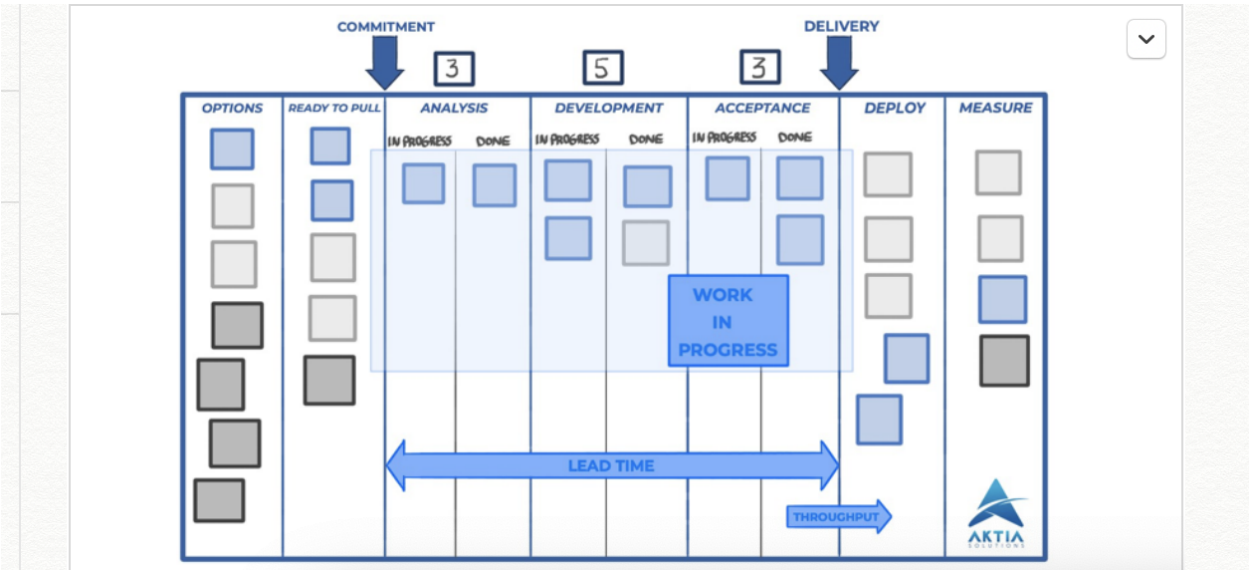



**Pull**
Implementing pull reduces WIP drastically and equilibrates demand to capacity. People need less supervision and meetings, because everybody knows what they have to do when they receive the signal.

**Cadence**
Working on a cadence forces batch size reduction, but it also has an important effect on transaction costs in the longer term

**The Core Kanban Practices**
1. Visualize
2. Limit work in progress
3. Manage flow
4. Make policies explicit
5. Implement feedback loops
6. Improve collaboratively, evolve experimentally

**Visualize**
A Kanban board such as in the image below is a typical way to visualize work and the process it goes through. For it to be a Kanban system rather than simply a board with stickers, the commitment and delivery points must be defined




Policies governing how we manage workflow are important to visualize too, for example by placing summaries between columns of the what must be done before items move from one column to the next.

Design the board; design the card info

Just a few things you might visualize for a proper Kanban System:
**Kanban System**
o Commitment Point and Delivery Point
o Definition of Ready (Commitment Point), Definition of Done (Delivery Point) and policies between columns
o Design of work item types o Process Steps / Activities
o In Progress vs Done States within process steps
o WIP Limits per column (process step)
o WIP Limits per swimlane (Classes of Service)
o Dependencies o Blockers Policy
o Classes of Service (CoS)
**Team performance:**
o Lead Time per Work Item Type or CoS
o Throughput per Work Item Time or CoS
o Predictability o Cumulative Flow Diagram
o Lead Time Histogram and Evolution
o Throughput Histogram and Evolution
o Lead Time Scatterplot
o OKRs

**Limit WIP**
What should be WIP limits ?

**Manage Flow**
We want to make sure we visualize everything and define the policies for managing the work. **We are not interested in who is doing what,** but in how fast the work is flowing through the system paying close attention to visual signals and controls.

**In Kanban we are obsessed about flow. We want work to flow as fast as possible to the customer with maximum quality and safety. This is an inheritance from Lean. We don't manage people; we manage the system to enable rapid flow.**

We must always keep an eye on queues, blockers and dependencies and make sure there are plenty of visual controls to warn us when something goes away.

**Make Policies Explicit**
Making things explicit is a key difference between high-performing teams and mediocre ones.

Humans tend to have a lot of assumptions and we cannot let assumptions and opinions drive businesses.

*WIP Limits* are one type of policy. Others include *capacity allocation*, *Definition of Done*, *Ready to Pul*l, or *replenishment policies* for the selection of new work when capacity is available. The use of *classes of service* is another policy example.

The team is responsible for defining the policies governing their work and they are also responsible for agreeing policies with interfaces upstream, downstream and other teams they depend on.

**Implement Feedback Loops**
We need to ensure mechanisms by which all levels across the organization are connected and flowing both strategically and operationally.

**Team Cadences**

**Daily Kanban**
All employees, regardless of their level, should be part of one daily meeting with their teams to discuss tactical issues, update their peers and synchronize. This is the usual daily huddle to plan for the next day and talk about **dependencies and blockers**.

**Replenishment**
It can occur on demand or within a specific frequency.

Typical activities for a replenishment meeting agenda are:
• Checking what's new since the last meeting
• **Inspect new arrivals to the pool of options/ideas**, and the "Ready to Pull" buffer
• Ensure that work items are correctly classified for **risk** and **classes of service** are appropriate.
• How many kanban slots are available and of what type/class?
• Filter the available pool of options by type, class and due date to gain an initial list of candidates for selection
• **Stakeholders must select a short list of candidates**

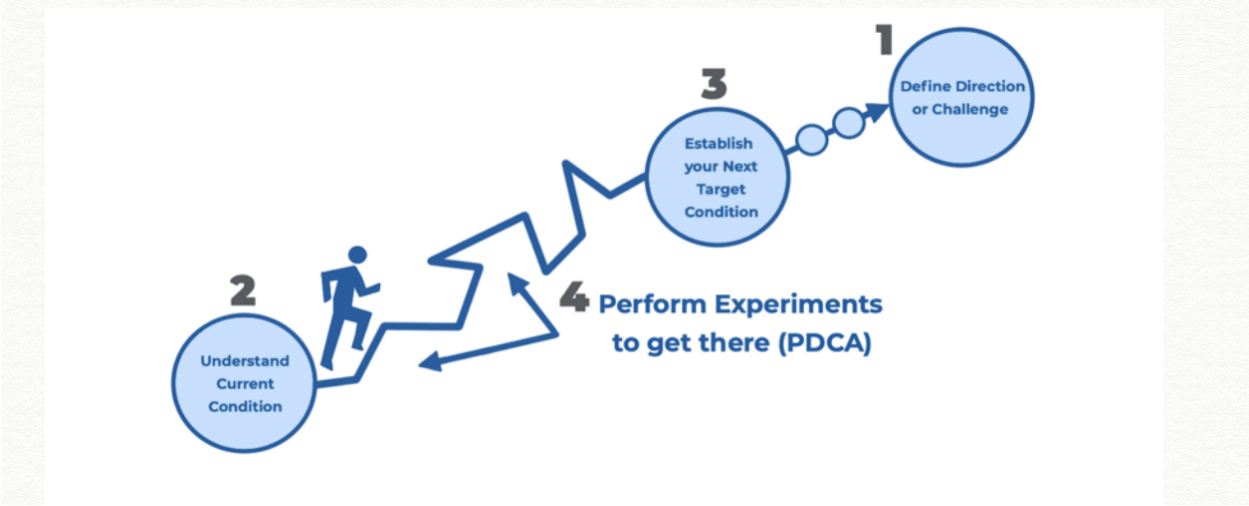
**Bi-weekly or Monthly Team Performance Meeting**
In this meeting you must review **Health Metrics** (process metrics), **Customer metrics** and **Product Metrics** to determine areas for improvement. In this meeting you must also review performance against **OKRs**.

**Delivery Planning Meeting**
Kanban *decouples planning from delivery*, so, unless you are doing Scrum, you must be able to plan delivery with downstream process or with customers.

• Which items will be ready for release?
• What is required to release into production?
• What testing will be required post-release?
• What metrics must be monitored post-release to ensure you achieved the expected outcomes of new features?
• What release risks are there? How are they being mitigated?
• What contingency plans are required?
• Who needs to be involved in the release?
• How long will the release take?
• What other logistics will be involved?

**Delivery Review Meeting**
If you are doing pure Kanban with single piece flow you need to establish a mechanism with your customers to be able to review your deliveries

**Improve collaboratively, evolve experimentally**
Design and test improvement experiments. Use your bi-weekly or monthly team performance meeting as the improvement cadence




**Kanban Board Glossary**
*Ready to Pull Policies*
A short checklist at the bottom of each column (exit criteria: things that should be done before the work can be moved to the next step.

*Classes of Service*
There are many different ways in which you can visualize classes of service that we will explore in our next book about Kanban System Design. For example:
• Different swim lanes on the board
• Visual indicator in the sticker
• Color of the sticker

*Explicit WIP Limits*
An explicit limit of work in progress. For the **whole board**, by **column** and/or by **swim lane**.

*Capacity Allocation*
An explicit limit of work in progress per work type, customer, lane, etc.

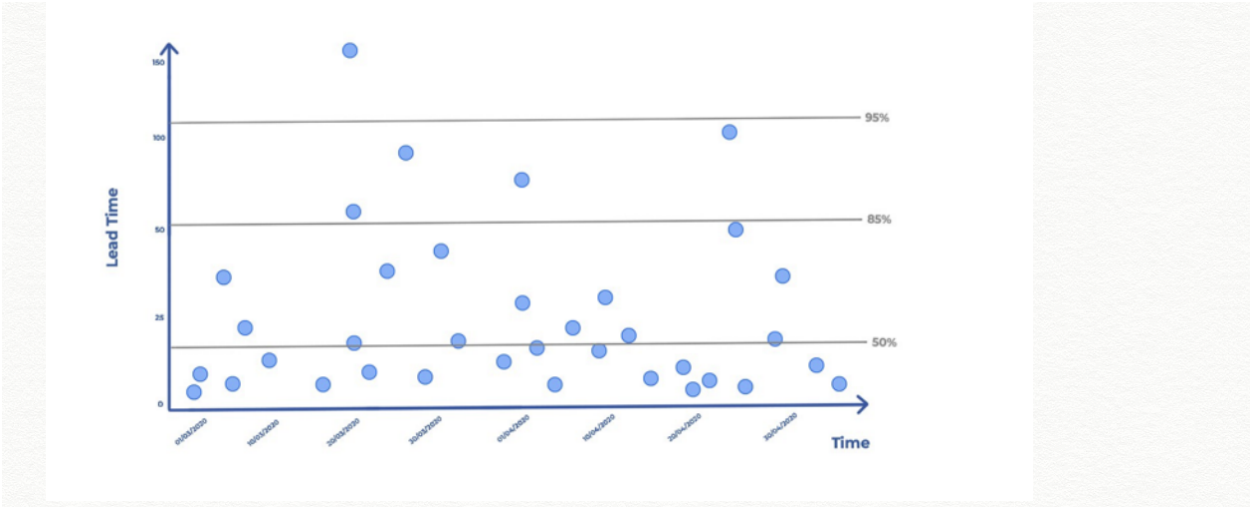
**How to improve Scrum with Kanban tools**
• Visualize dependencies. Especially long-term dependencies and define policies for managing them
• Visualize blocked items with proper information and their associated unblocker sticker
• Visualize WIP limits
• Visualize **classes of service**
• Visualize Ready to Pull policies
• Visualize Definition of Ready and Definition of Done
• Visualize upstream process for elaborating the items in the backlog
• Visualize downstream process after an item has departed from the team
• Visualize performance metrics

**Blocker Analysis**
There is fundamental practice in Lean called *Jidoka*, which allows anyone in the value chain to stop the process if an abnormality is found that harms the quality of the product.

Gather a few of those problems, separate them between internal and external problems, group similar ones and do root cause analysis for the cluster so that a long-term fix can be implemented.

**Outlier Analysis**

You can analyze outliers and see if they are caused by common cause or special cause variability. Some other learnings that might come from studying outliers are "Oh, we forgot to move the ticket! What's going on here?". If the team cannot trust their own metrics it is going to be difficult to improve




**Make Policies Explicit**
Many Scrum teams work based on a lot of assumptions and wishful thinking that could really benefit by just making this practice real.

Example. Discuss with your team what is the criteria for prioritizing product backlog items and then make a policy out of it.

**Implement Feedback Loops**
Include a little more science and data into your inspection and adaptation cycles.

**Improve collaboratively, evolve experimentally**
The improvements coming out of retrospectives sometimes seem random, superficial and not aligned with team's or organization's goals.

In the same way that when we choose items from the backlog, we do it to achieve some specific outcomes, when we choose problems to solve, we should choose those that bring us closer to our improvement objectives.

**After learning the basics of flow and Kanban systems, now you are ready to design your own Kanban System.**

The process has the following ten steps:
1. Select a service
2. Define service's mission
3. Analyze sources of dissatisfaction
4. Discover what the service does

5. Identify work types
6. Define visualization policies
7. Create and populate the board
8. Define way-of-working policies
9. Define WIP policies
10. Define meeting policies

**Define Mission**
A mission must be a short statement describing the reason an organization, team or service exists.

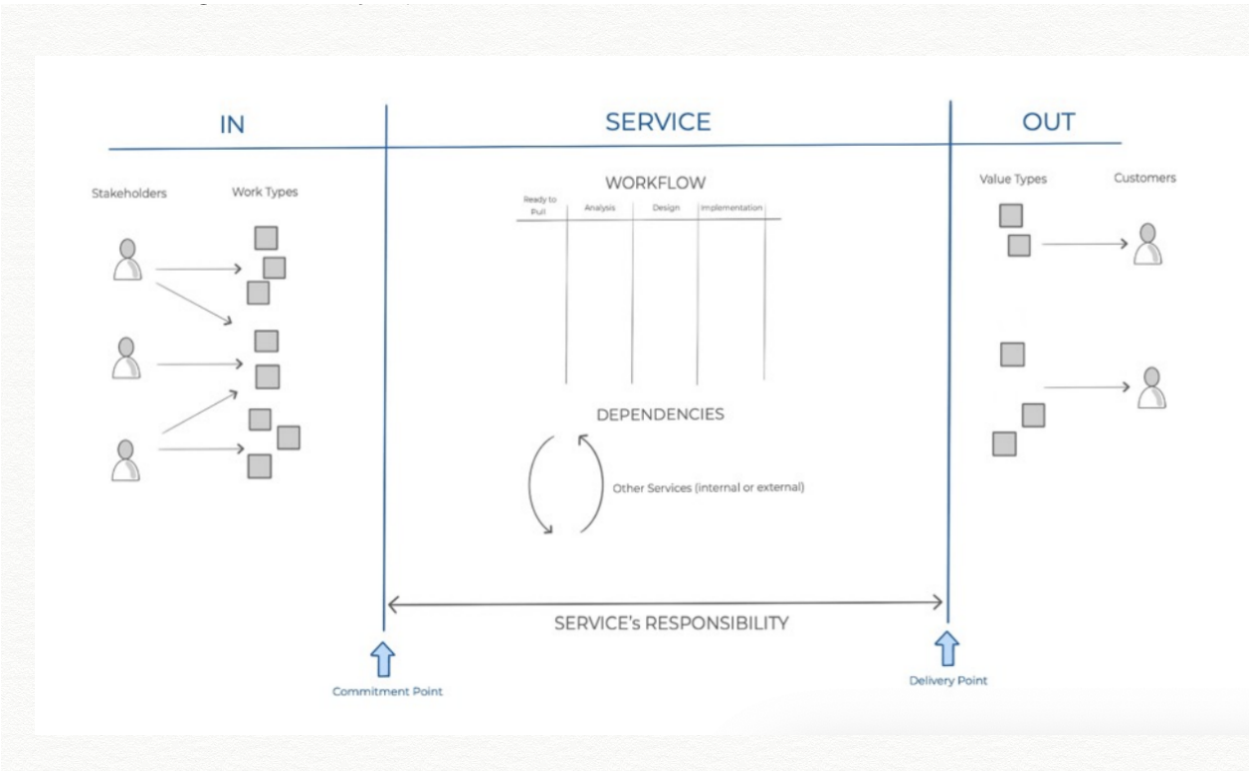
**Analyze Sources of Dissatisfaction**

The purpose of this step is to surface any major dissatisfactions with your team's current situation.

As part of this process we will also need to bring into the conversation customer's and stakeholder's dissatisfactions with the service and see if there are any direct matches.

**Discover What the Service Does**

This step entails four stages:
1. **Identify stakeholders and the type of work** they request to the team. Remember stakeholders can be customers, the team itself, other teams within the company or business stakeholders.
2. **Identify the outcome** of our service. In most of the cases the type of work getting out of the Kanban System is the same being requested, but that's not always the case; in some cases we might be delivering a partial work to another team who finalizes the work (think of operations teams in IT organizations) or we can be delivering the work but someone else delivers it to the end customer (think of an eCommerce and the last-mile delivery company they use)
3. Jump into the team's area of responsibility and model the workflow. It is very important here that we model what the team is currently doing, not what you think you are doing, or you would like to be doing
4. The last stage is to identify dependencies with other services (internal or external)



**Identify Work Types**

Everyone dumps into stickies everything they do. Once everyone has downloaded their brains they can start grouping and categorizing work.

Once you have gathered and grouped all sorts of work the service gets requested, you must analyze *demand* for each work item type and make sure you classify them based on the following factors: *most valuable, most urgent, largest quantity of demand, smallest quantity of demand and alignment with team's mission*.

As part of this step it is key that you get a clear idea of how the work arrives into the team: *by email, conversations, tools.* And, in what frequency: *stable regular cadence, in random bursts, seasonally*.

**Visualization Policies**

To the question, "*what should we visualize?*" I would recommend visualizing "*whatever will help you make better collective decisions*".
Can be:
• Work Types
• Work Item Design
• Workflow
• What work not to visualize
• How often must the visualization be in sync

**Create and Populate the Board**

We ask the team to draw the dashboard and write in stickers all the work currently in progress, as well as all work in Ready to Pull column, by following the policies defined in the previous step.

We also request the team to define other important policies like Definition of Ready (DoR) and Definition of Done (DoD).

**Create & Populate the Board**
1. Draw the workflow as agreed in step 4.
2. Create work items
3. Populate the board

**Make Sure Work Items Are at The Right Place**
1. Create a Definition of Done for each column
2. Create Ready to Pull columns
3. Tag blocked items

**Prioritization Policies for Work in Progress**
1. Dues Dates
2. Prioritize ongoing work
3. Prioritization policy
4. Definition of Ready

**Define Way-of-Working Policies**

**Demand Management**
Discuss with the team who is managing the *Ready to Pull* buffer or the upstream process in case of an end-to-end product team.

**Side Orders (AKA Dark Matter)**

The team must agree on how to handle demands coming “from the side” (i.e. not via an official channel: chat, telephone calls, “corridor” orders, etc.). And also, what to do with that work some team members are *unofficially* doing.

**When to Pull**
A good policy would state that: a team member should pull new work only when he/she cannot contribute to any other ongoing work. If you are using WIP limits, the policy should state that work is pulled only if the WIP limit is not violated.

An excellent policy would also say that instead of violating the WIP limit and pulling new work, a team member should consider working on intangible work (e.g. improving test environment, compilation scripts, documentation, refactoring, etc.), or helping another team member or supporting the Product Owner in developing upstream options.

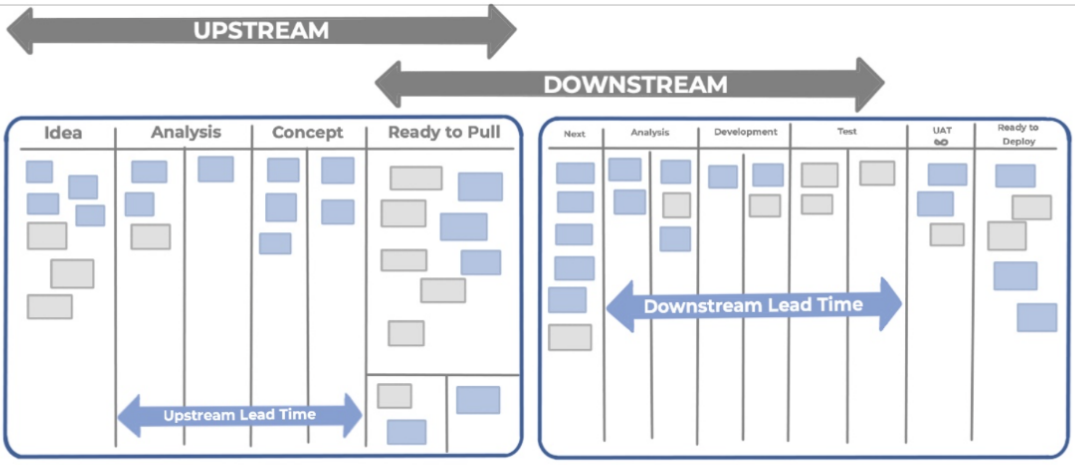
**Define WIP Policies**
Together with the visualization policies, the team must agree on what happens in certain situations like when policies are violated, or when there are long-term blockers, when in-process defects arise or when an important stakeholder asks the team to expedite some work.

**Define Meeting Policies**

For most teams that start with Kanban, they must at least design the five cadences we talked about in previous chapters:
**Getting things done:**
• Daily Kanban (daily)
• Delivery Planning Meeting (per delivery)
**Doing the right things:**
• Replenishment (weekly, on-demand)
• Strategy Review (quarterly)
• *Delivery Review Meeting*
**Doing things better:**

- Operations Review (monthly) - **This is a monthly review of demand and capability for each Kanban System in the organization with a particular focus on dependencies. It provides systemic overview as everybody brings updated performance data and dependencies from their Service Delivery Review meetings.**
- Risk Review (monthly)
- Service Delivery Review (bi-weekly) - **Bi-weekly Kanban feedback loop where we look at whether we are delivering according to customer expectations.**
- Team Performance Meeting

**Upstream vs Downstream**

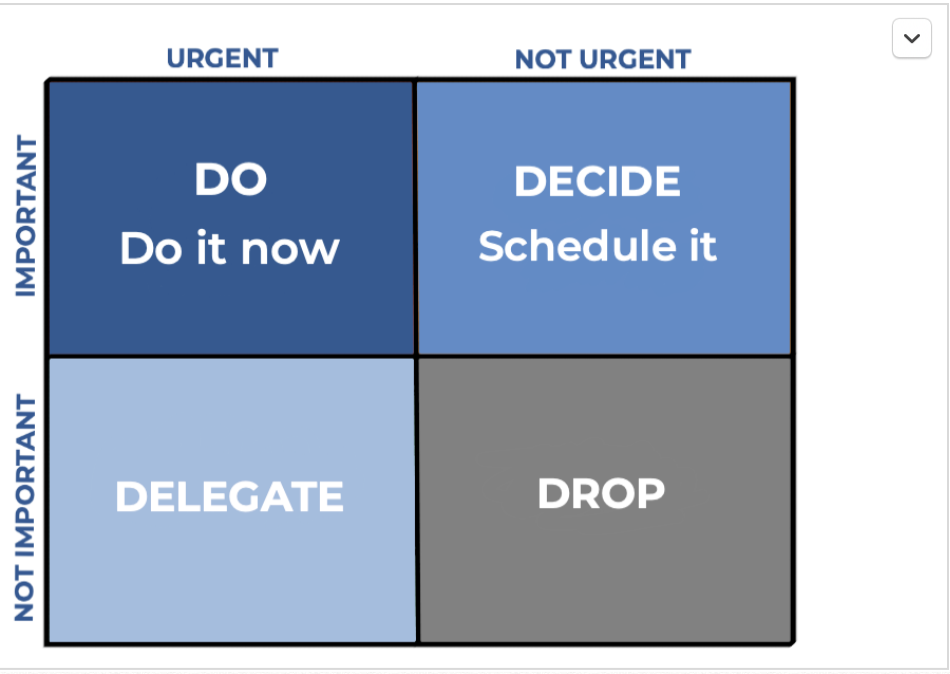



In this example, we can see two Kanban Systems visualizing a typical end-to-end software development flow starting from the capture of ideas/options and ending in work items that are ready to be deployed:

- Downstream (or Delivery), managed by the delivery team
- Upstream, managed all together by the different stakeholders who request work to the team


- Downstream (or Delivery), managed by the delivery team
- Upstream, managed all together by the different stakeholders who request work to the team


**Upstream Kanban is essentially a Triage Process**



many companies and teams still have difficulties in prioritzing work because they lack all the basic information and decision frameworks to be able to do that quickly and effectively.

First question you must be able to answer is *“Should we work on this at all?”*. Second is *“When?”*. And, with upstream Kanban we add another question: *“Following which workflow?”*.

**Designing Upstream Kanban for Product Teams**