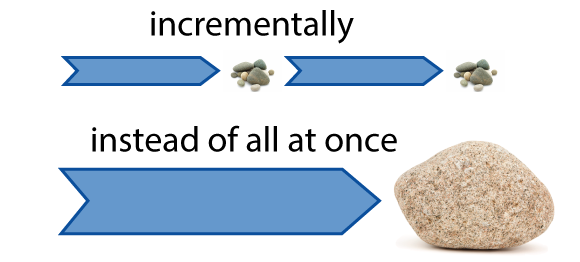
**BY – SHANTANU RANA**

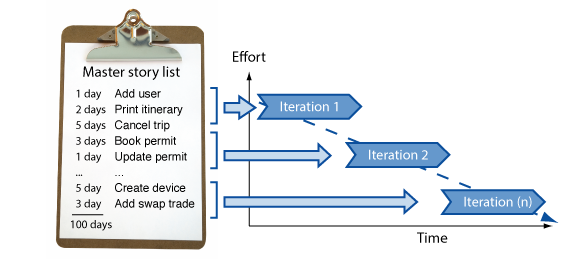
**What is Agile?**

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short 2 weeks cycles called sprints(iterations).



It works by breaking projects down into little bits of user functionality called [user stories](http://www.agilenutshell.com/user_stories), prioritizing them, and then continuously delivering them in short two week cycles called [iterations](http://www.agilenutshell.com/iterations).



# How does it work?

At its core, Agile does the same thing you and I do when faced with too much to do and not enough time.

### You make a list

Sitting down with your customer you make a list of features they would like to see in their software. We call these things [user stories](http://www.agilenutshell.com/user_stories) and they become the To Do list for your project.



### You size things up

Then, using Agile [estimation](http://www.agilenutshell.com/estimation) techniques, you size your stories relatively to each other, coming up with a guess as to how long you think each user story will take.



### You set some priorities

Like most lists, there always seems to be more to do than time allows. So you ask your customer to prioritize their list so you get the most important stuff done first, and save the least important for last.



### You start executing

Then you start delivering some value. You start at the top. Work your way to the bottom. Building, iterating, and getting feedback from your customer as you go.

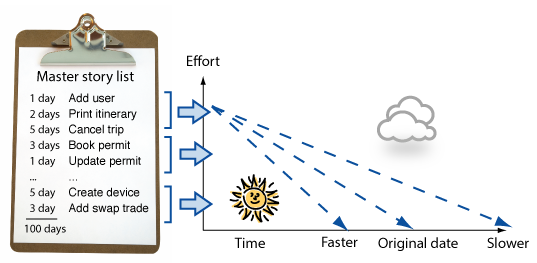


### You update the plan as you go.

Then, as you and your customer starting delivering, one of two things is going to happen. You'll discover:

1. You're going fast enough. All is good. Or,
2. You have too much to do and not enough time.

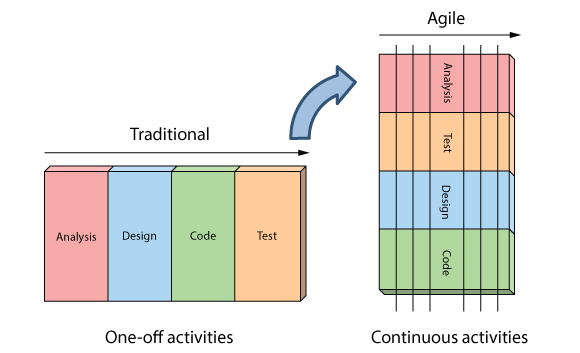
At this point you have two choices. You can either a) do less and cut scope (recommended). Or you can b) push out the date and ask for more money.



# **How is Agile different?**

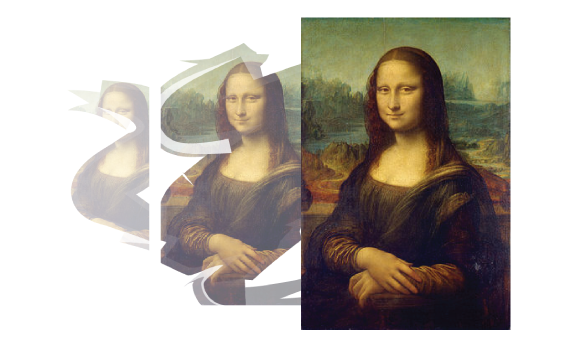
### Analysis, design, coding, and testing are continuous activities

You are never done analysis, design, coding and testing on an Agile project. So long as there are features to build, and the means to deliver them, these activities continue for the duration of the project.



### Development is iterative

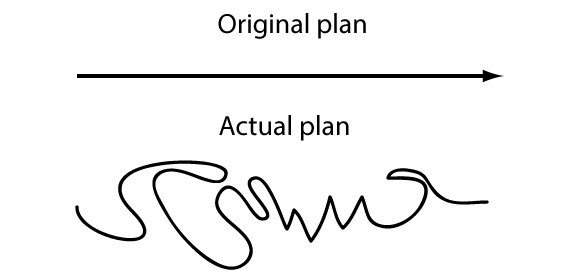
Iterative development means starting with something really simple, and adding to it incrementally over time.



It means evolving the architecture, accepting that your requirements are going to change, and continuously refining and tweaking your product as you go.

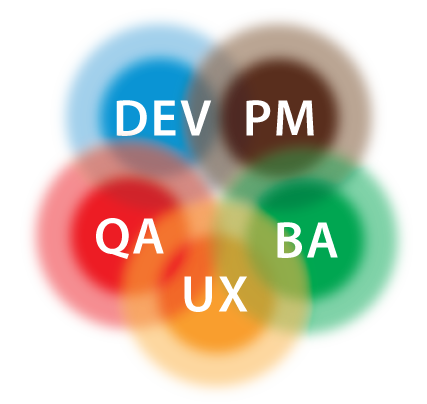
### Planning is adaptive

When reality disagrees with their plans, Agilists find it easier to change their plans than reality. They call this adaptive planning.



And while there are many ways to changes plans, the preferred way is to flex on scope.

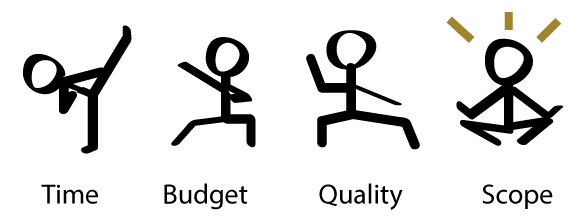
### Roles blur



Roles really blur on Agile projects. When it’s done right, joining an Agile team is a lot like working in a mini-startup. People pitch in and do whatever it takes to make the project successful—regardless of title or role.

Yes, people still have core competencies, and, yes, they generally stick to what they are good at. But on an agile project, narrowly defined roles like analyst, programmer, and tester don’t really exist - at least not in the traditional sense.

### Scope can vary

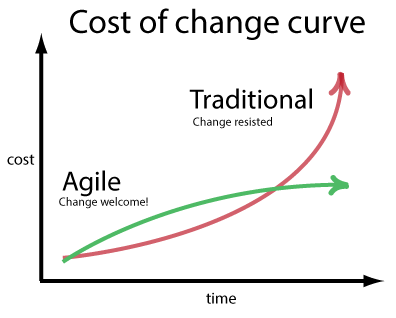


Agile deals with the age old problem of having too much to do and not enough time by doing less.

By fixing time, budget, and quality, and being flexing around scope, Agile teams maintain the integrity of their plans, work within their means, and avoid the burn out, drama, and dysfunction traditionally associated with our industry.

### Requirements can change

Traditionally change has been shunned on software projects because of it's high perceived cost late in the game. Agile challenges this notion and believes the cost of change can be relatively flat.



Through a combination of modern software engineering practices, and open and honest planning, Agilsts accept and embrace change even late in delivery process.

### Working software is the primary measure of success



The rate at which teams can turn their customer's wishes into working software is how Agilists measure productivity. Project plans, test plans, and analysis artifacts are all well and good but Agilists understand they in themselves are of no value to the end customer.

**Principles of Agile**

1

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2

Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4

Business people and developers must work together daily throughout the project.

5

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7

Working software is the primary measure of progress.

8

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9

Continuous attention to technical excellence and good design enhances agility.

10

Simplicity–the art of maximizing the amount of work not done–is essential.

11

The best architectures, requirements, and designs emerge from self-organizing teams.

12

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

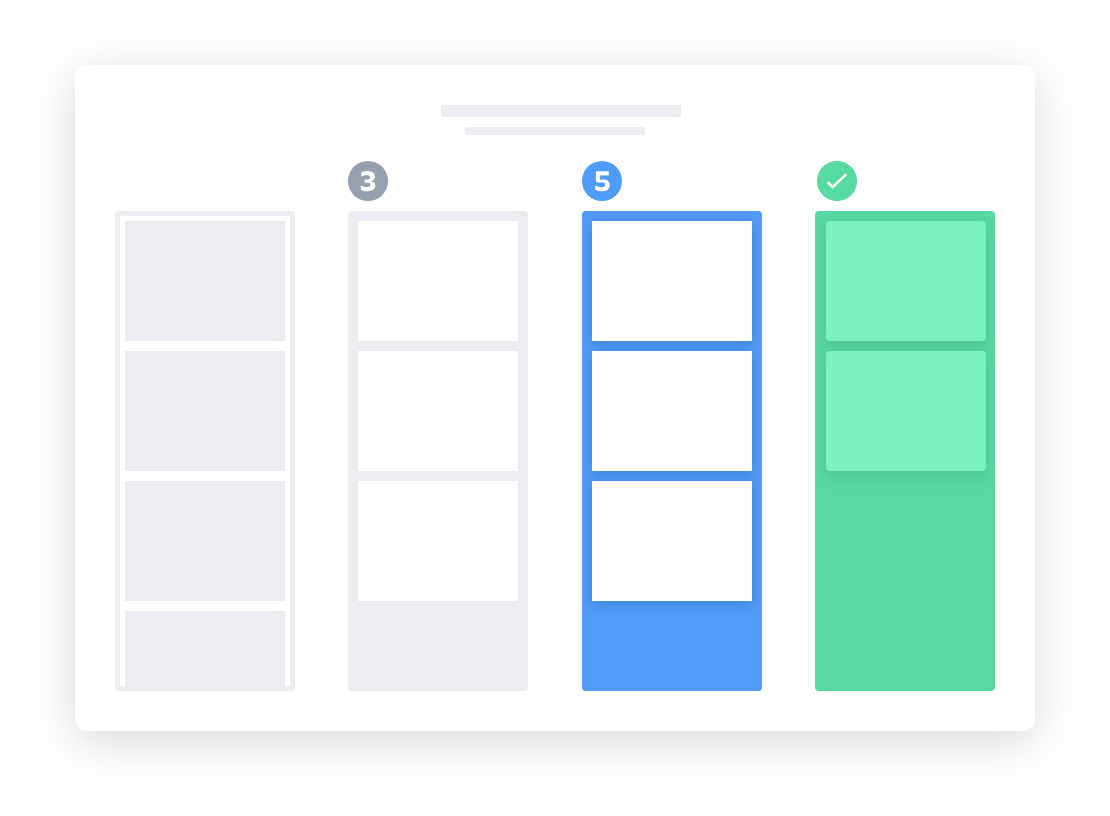
# **What is a kanban board?**

A kanban board is an agile project management tool designed to help visualize work, limit work-in-progress, and maximize efficiency (or flow). It can help both [agile](https://www.atlassian.com/agile) and [DevOps](https://www.atlassian.com/devops/what-is-devops) teams establish order in their daily work. Kanban boards use cards, columns, and continuous improvement to help technology and service teams commit to the right amount of work, and get it done!

Youtube link = > https://youtu.be/Bcid33tgq8A

["Kanban" is the Japanese word for "visual signal." If you work in services or technology, your work is often times invisible and intangible. A kanban board helps make your work visible so you can show it to others and keep everyone on the same page.](https://twitter.com/intent/tweet?text=%20%20%22Kanban%22%20is%20the%20Japanese%20word%20for%20%22visual%20signal.%22%20If%20you%20work%20in%20services%20or%20technology,%20your%20work%20is%20often%20times%20invisible%20and%20intangible.%20A%20kanban%20board%20helps%20make%20your%20work%20visible%20so%20you%20can%20show%20it%20to%20others%20and%20keep%20everyone%20on%20the%20same%20page.%20&url=https://www.atlassian.com/agile/kanban/boards&via=Atlassian)

[Kanban](https://www.atlassian.com/agile/kanban) has come a long way from its origins in lean manufacturing thanks to a small but mighty group of kanban enthusiasts. David Anderson’s work defining the [kanban method](https://djaa.com/principles-and-general-practices-of-the-kanban-method/) helped bring kanban into the software and services space, and [Personal Kanban](http://personalkanban.com/pk/), by Jim Benson and Tonianne DeMaria, helped expand the applications of kanban to places you wouldn’t believe.

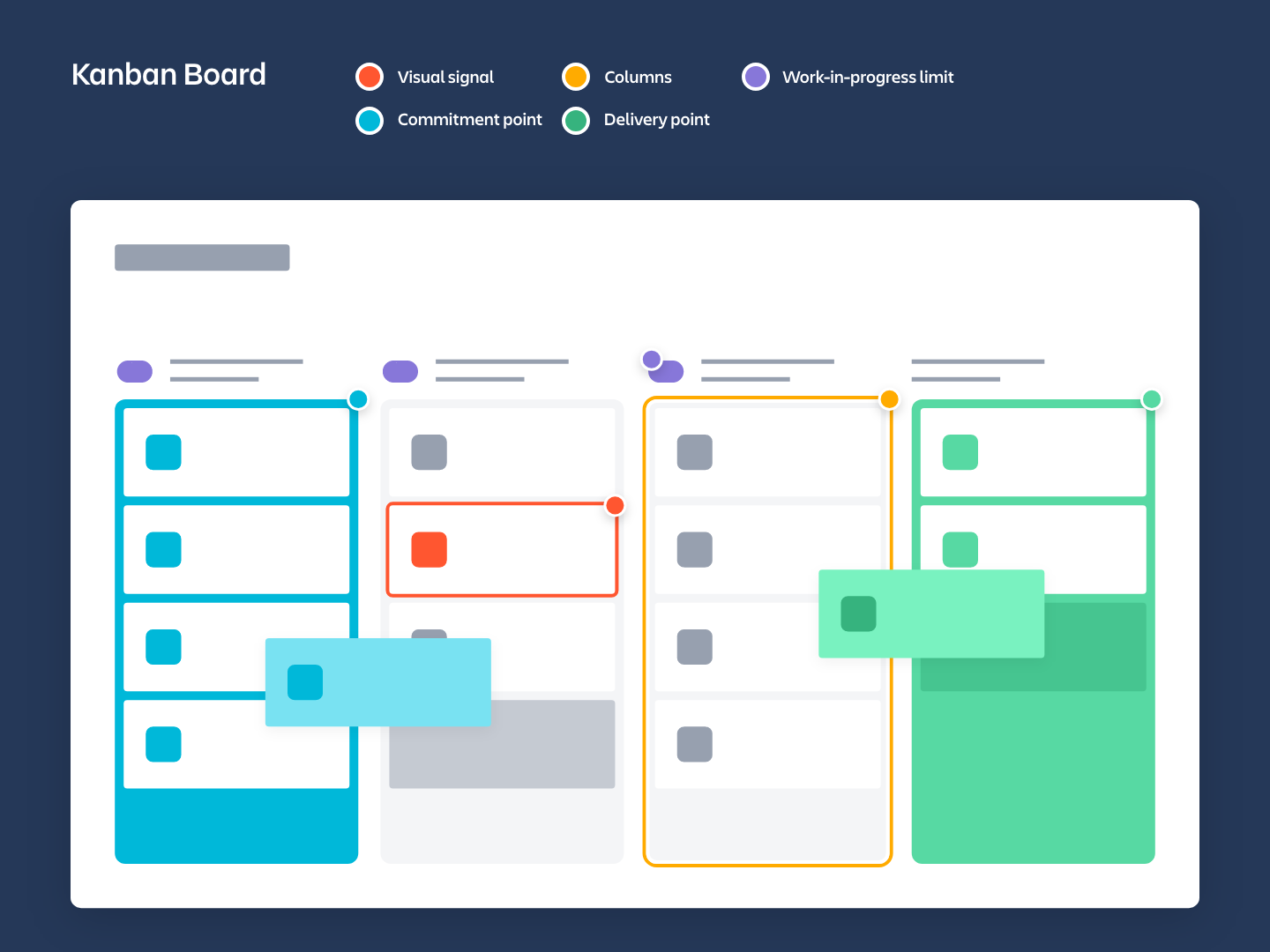


I use kanban boards every day and couldn’t imagine life without them. The ideas and best practices here are a peppering of my personal experiences, research, and conversations I had with with [Zach Nies](https://twitter.com/zachnies), [Keith Nottinson](https://twitter.com/keithadam), and [Jim Benson](https://twitter.com/ourfounder).

What keeps me coming back to kanban are the kanban values and (surprising) lack of rules. The kanban values are respect for people and continuous improvement.

## Elements of a kanban board

David Anderson established that kanban boards can be broken down into five components: Visual signals, columns, work-in-progress limits, a commitment point, and a delivery point.



1. **Visual Signals** — One of the first things you’ll notice about a kanban board are the visual cards (stickies, tickets, or otherwise). Kanban teams write all of their projects and work items onto cards, usually one per card. For agile teams, each card could encapsulate one[user story](https://www.atlassian.com/agile/project-management/user-stories). Once on the board, these visual signals help teammates and stakeholders quickly understand what the team is working on.
2. **Columns** — Another hallmark of the kanban board are the columns. Each column represents a specific activity that together compose a “workflow”. Cards flow through the workflow until completion. [Workflows](https://www.atlassian.com/agile/project-management/workflow) can be as simple as “To Do,” “In Progress,” “Complete,” or much more complex.
3. **Work In Progress (WIP) Limits** — WIP limits are the maximum number of cards that can be in one column at any given time. A column with a WIP limit of three cannot have more than three cards in it. When the column is “maxed-out” the team needs to swarm on those cards and move them forward before new cards can move into that stage of the workflow. These WIP limits are critical for exposing bottlenecks in the workflow and maximizing flow. WIP limits give you an early warning sign that you committed to too much work.
4. **Commitment point** — Kanban teams often have a backlog for their board. This is where customers and teammates put ideas for projects that the team can pick up when they are ready. The commitment point is the moment when an idea is picked up by the team and work starts on the project.
5. **Delivery point** — The delivery point is the end of a kanban team’s workflow. For most teams, the delivery point is when the product or service is in the hands of the customer. The team’s goal is to take cards from the commitment point to the delivery point as fast as possible. The elapsed time between the two is the called Lead Time. Kanban teams are continuously improving to decrease their lead time as much as possible.

A kanban board with these five elements will undoubtedly set your team up for success. But now, I’ll present an opposing point of view.

Jim Benson says that kanban only has two rules: Limit work in progress and visualize your work. If you start with just those rules and apply them to your work, your kanban board will look much different than the one described above. And thats ok! Jim advocates for starting with just these two rules because, he says, “The more rules you add, the less contexts it fits into.”