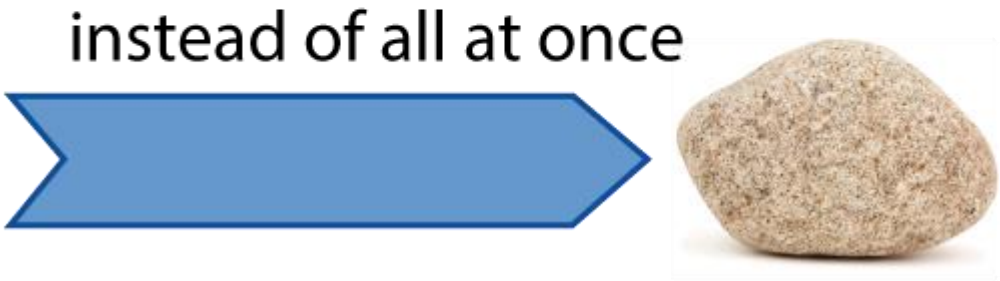
BY – SHANTANU RANA

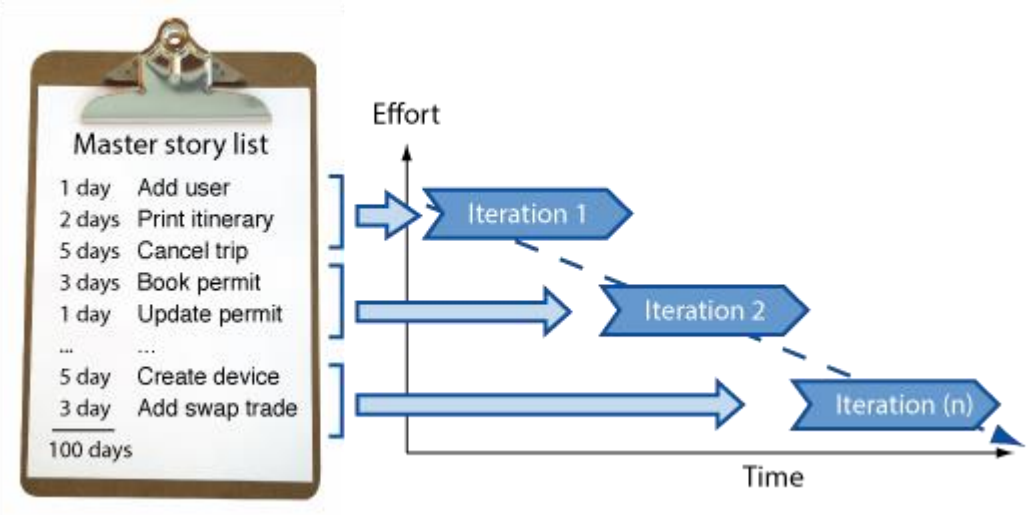
What is Agile?

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, prioritizing them, and then continuously delivering them in short two week cycles called 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 and they become the To Do list for your project.

You size things up

Then, using Agile 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

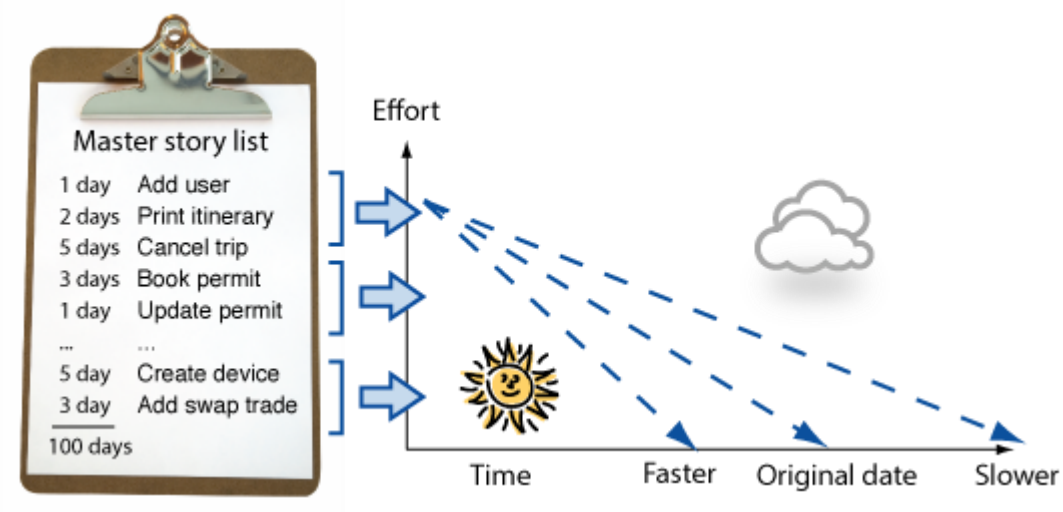
Then you start delivering some value. You start at the top. Work your way to the bottom. Building, iterating, and getting feedback from your customer as you go.

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:

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

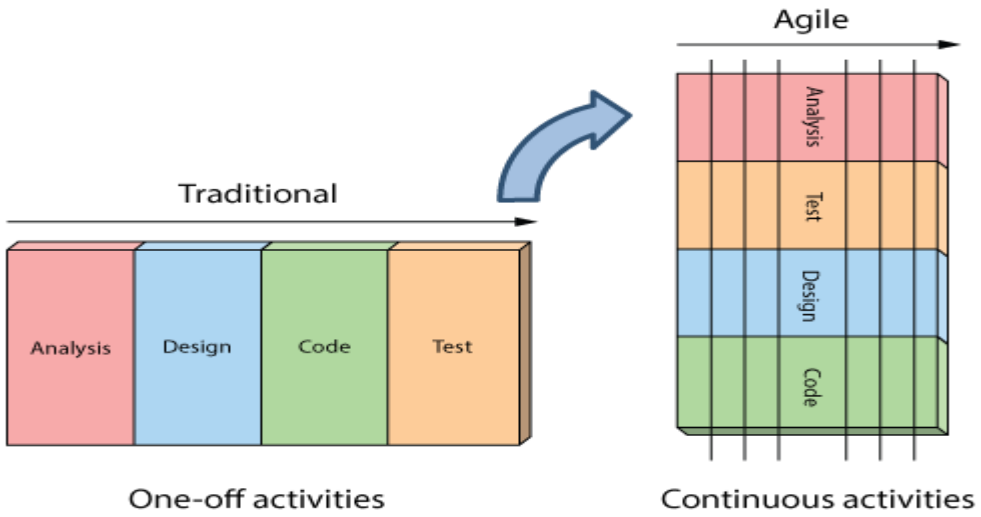
At this point you have two choices. You can either a) do less and cut scope (recommended). Or you can b) push out the date and ask for more money.



How is Agile different?

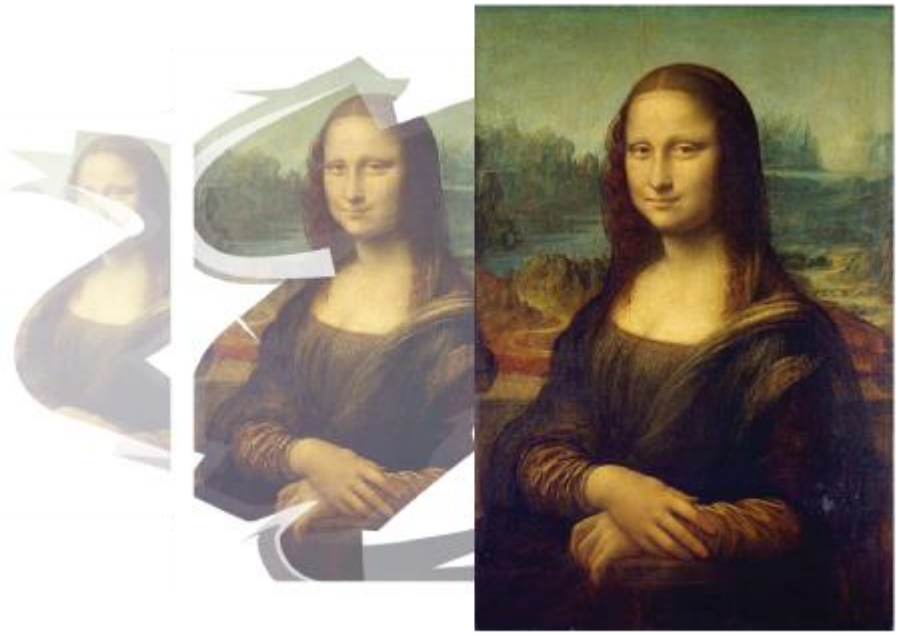
1. Analysis, design, coding, and testing are continuous activities

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.



2. Development is iterative

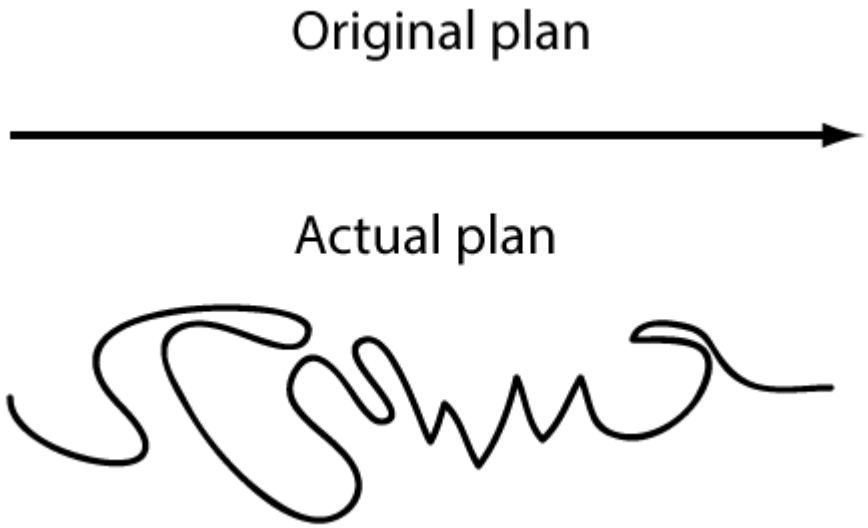
Iterative development means starting with something really simple, and adding to it incrementally over time.



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

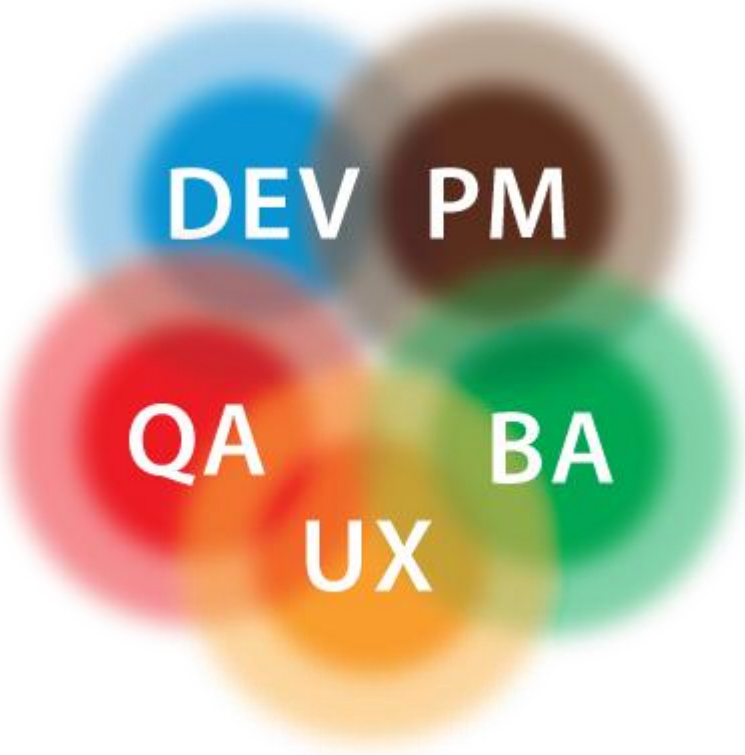
3. Planning is adaptive

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.

4. 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.

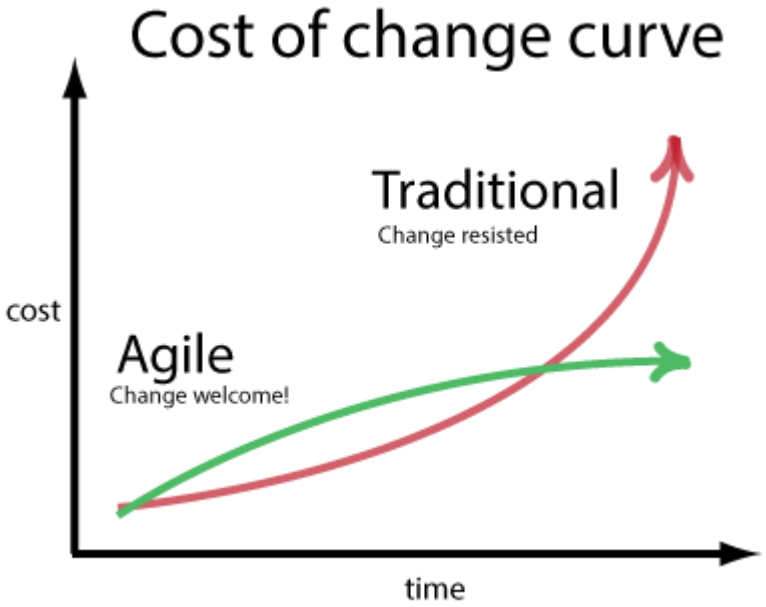
5. Scope can vary

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.

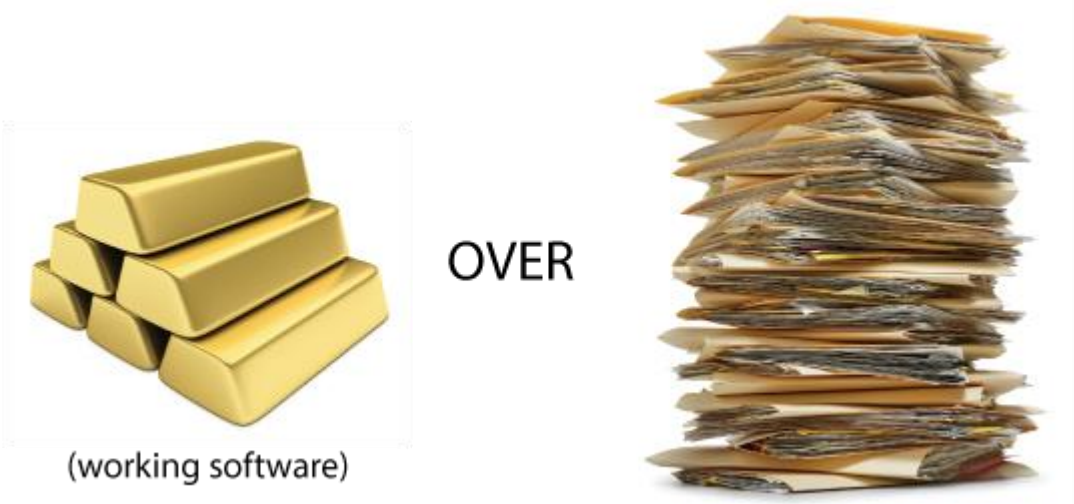
6. Requirements can change

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.

7. 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.