Overview of

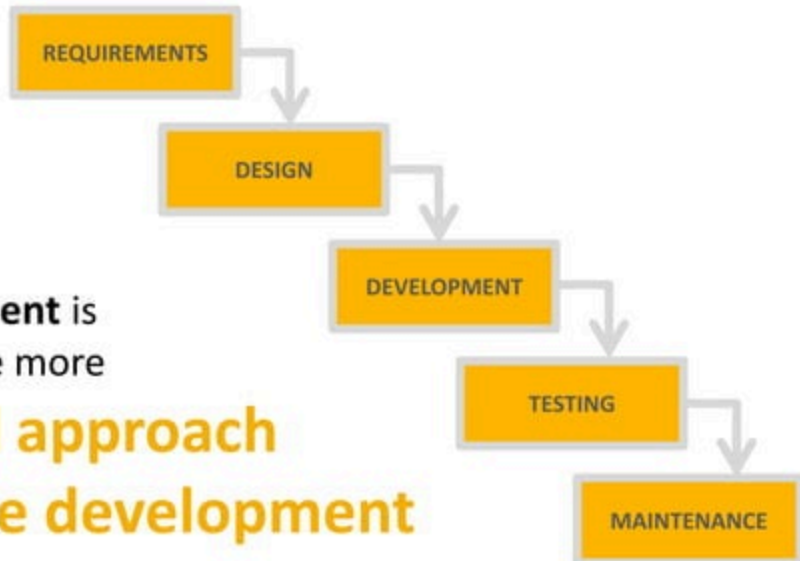
# Agile Methodology

Prepared by: Haresh Karkar [Information Architect]

A [really] short history of

# Software development processes

# Waterfall Development




**Waterfall Development** is another name for the more

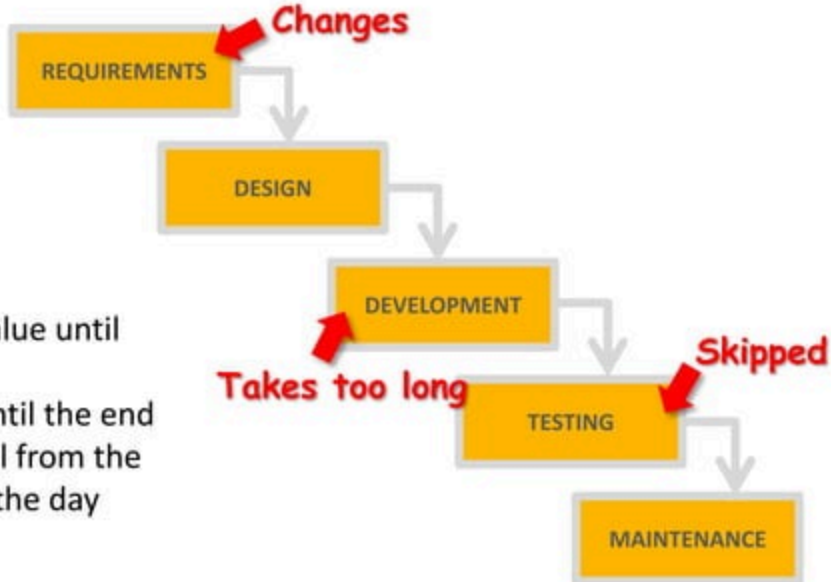
**traditional approach to software development**

# Waterfall Development (contd..)

You **complete one phase** (e.g. design) **before** moving on to the **next phase** (e.g. development)

You **rarely aim to re-visit a 'phase' once it's completed**. That means, you **better get whatever you're doing right the first time!**

# But...




- You don't realize any value until the end of the project
- You leave the testing until the end
- You don't seek approval from the stakeholders until late in the day

This approach is **highly risky**, often more **costly** and generally **less efficient** than **Agile** approaches


**Not a process**, it's a **philosophy** or **set of values**

# Agile Manifesto

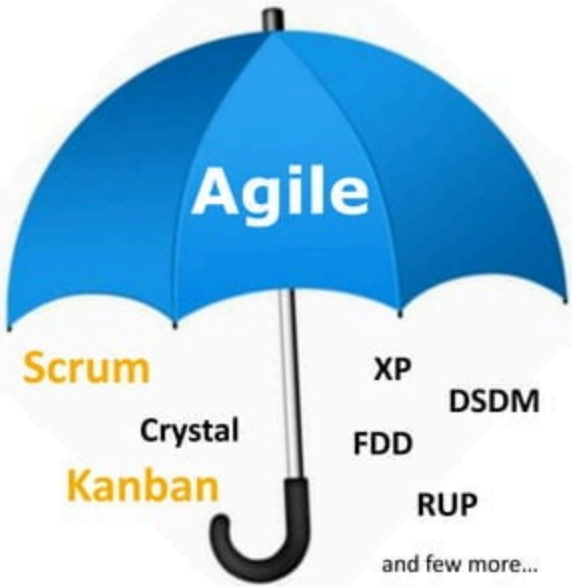
**Individuals and interactions** over **processes and tools**

**Working software** over **comprehensive documentation**

**Customer collaboration** over **contract negotiation**

**Responding to change** over **following a plan**

# Agile Umbrella




* Check wikipedia for list of all Agile methods

More Prescriptive
**more rules** to follow

- RUP (120+)
  RUP has over 30 roles, over 20 activities, and over 70 artifacts
- XP (13)
- Scrum (9)
- Kanban (3)
- Do Whatever!! (0)

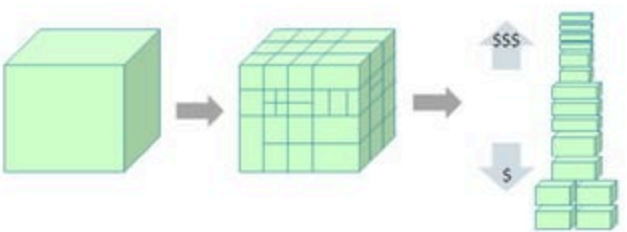
More Adaptive
**fewer rules** to follow

# A **light-weight agile process tool**

# Scrum

**Split your organization**
into small, cross-functional, self-organizing teams.


**Split your work** into a list of small, concrete deliverables. Sort the list by priority and estimate the relative effort of each item.



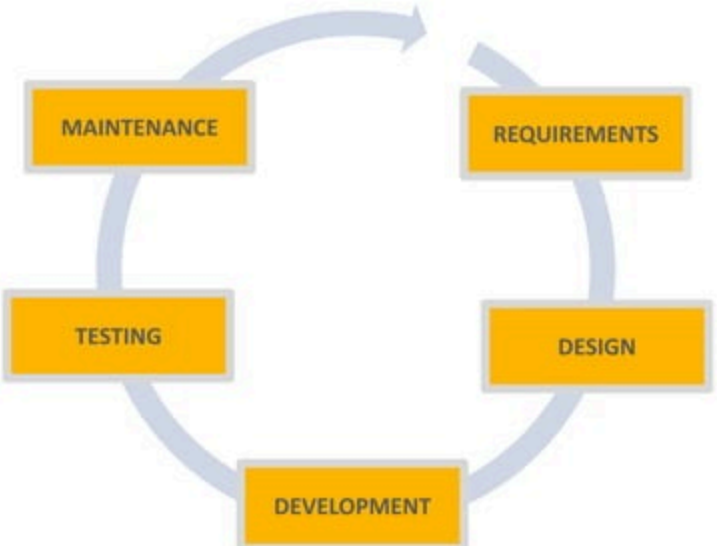
# Scrum (contd..)

**Split time** into short fixed-length iterations/ sprints (usually 2 – 4 weeks), with potentially shippable code demonstrated after each iteration.

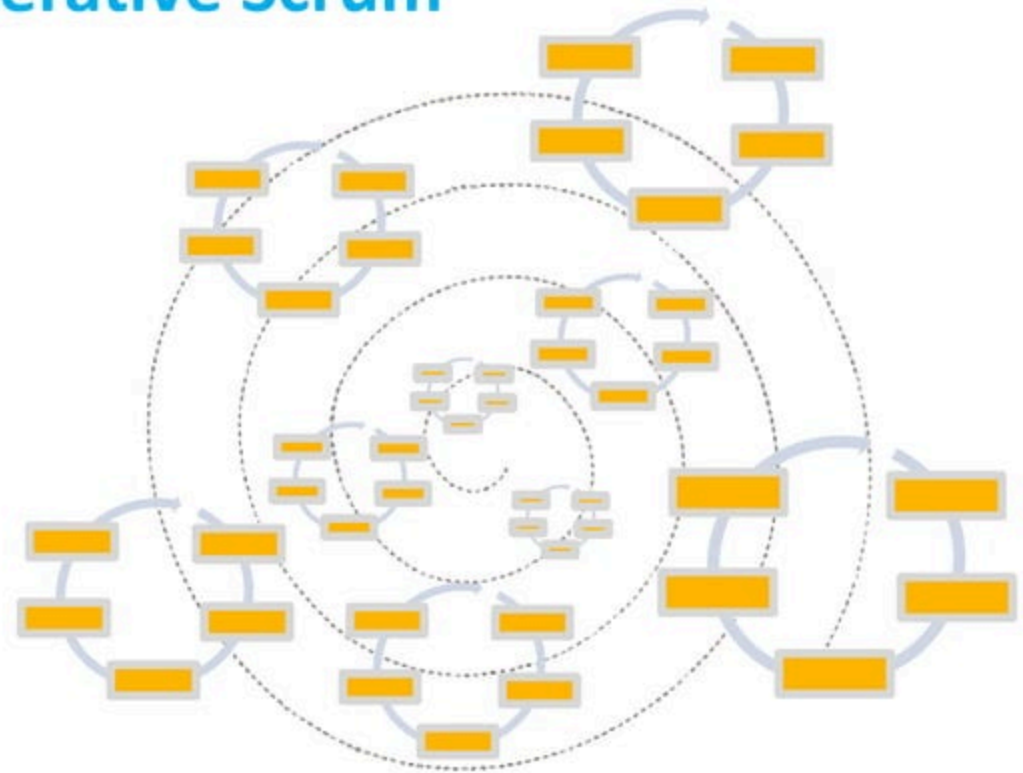

**Optimize the release plan** and update priorities in collaboration with the customer, based on insights gained by inspecting the release after each iteration.

**Optimize the process** by having a retrospective after each iteration.

# Scrum vs. Waterfall



# Iterative Scrum



# Things we do in Scrum

a.k.a Scrum terminologies

The project/ product is described as a list of features: the **backlog**.

The features are described in terms of **user stories**.

The scrum team **estimates** the **work** associated with each story.

Features in the backlog are **ranked** in order of importance.

**Result:** a **ranked** and **weighted** list of product features, a **roadmap**.

**Daily scrum meeting** to discuss **What did you do y'day**? **What will you do today**? **Any obstacles**?

# Scrum Artifacts

Sample Userstory

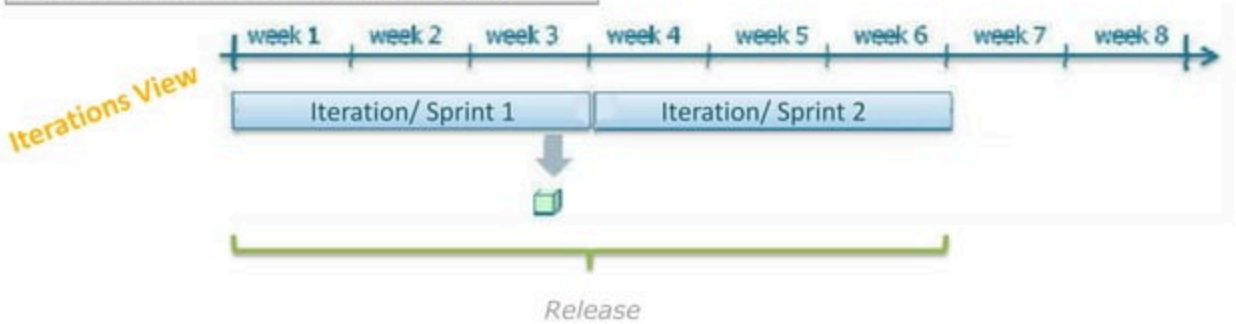
USERS SHOULD BE ABLE TO UPLOAD MULTIPLE PHOTOS AT ONCE

- Test with JPEG, PNG, GIF (supported)
- Test with an unsupported image format
- Test with Flash not present
- Test with more than 20 MB of total POST data

Efforts 10hrs

Efforts: 2hrs IA, 6hrs Development, 2hrs Testing

The total effort each iteration can accommodate **leads to** number of user story per iteration




One **release** may contains **number of iterations**

# Scrum planning example

**Iteration cycle** of **3 weeks**

**Working hours per day** is **8**

Total hours of **work iteration can accommodate**

8hrs x 5days x 3weeks = **120hrs**

**Product backlog** of **20 stories**

**Each story effort** is **10 hrs**

---

**Iteration backlog** or **number of stories per iteration**

**12 user story**

# Scrum in a nutshell

So instead of a **large group** spending **a long time** building a **big thing**, we have a **small team** spending a **short time** building a **small thing**.

**But integrating regularly to see the whole.**


Limit Work-In-Progress
Visualize the
Visual Card
KANBAN
Work
Signboard
Measure & Manage Flow
Just-in-time (JIT)

## Lean approach to agile development Kanban

Similar to Scrum in the sense that you **focus on features as opposed to groups of features** – however Lean takes this one step further again.

> You **select, plan, develop, test and deploy one feature** (in its simplest form) **before you select, plan, develop, test and deploy the next feature**.

**Aim** is to **eliminate 'waste'** wherever possible…

# Kanban (contd...)

## Visualize the workflow

- Split the work into pieces, write each item on a card and put on the wall
- Use named columns to illustrate where each item is in the workflow

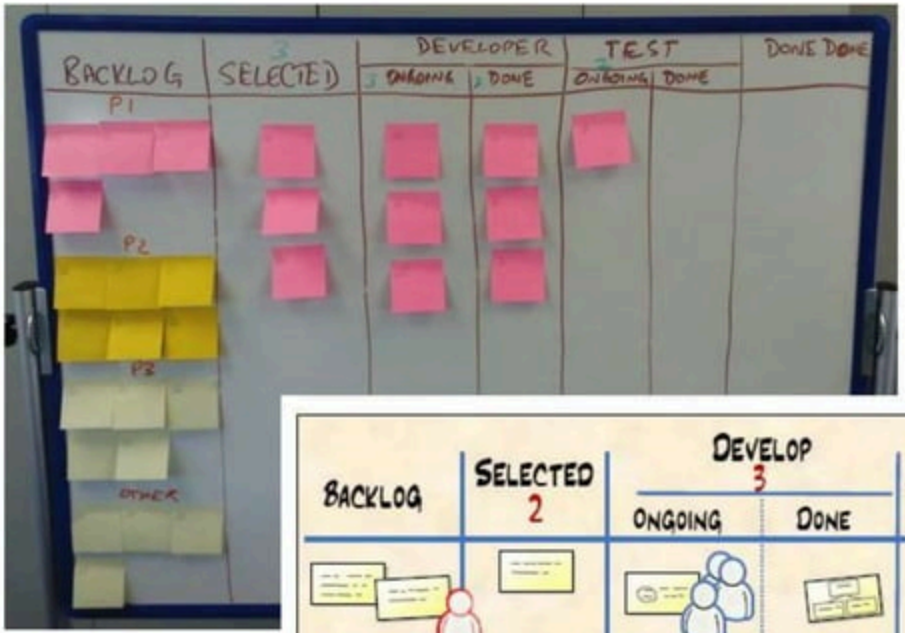

## Limit WIP (work in progress)

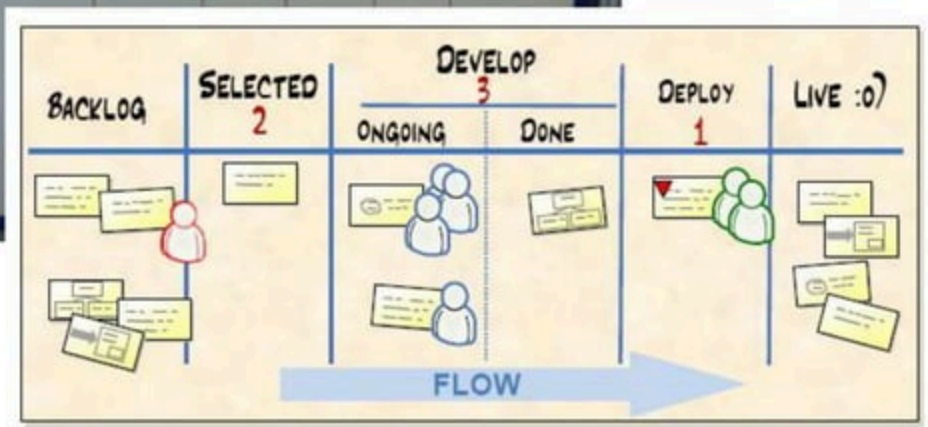
- Assign explicit limits to how many items may be in progress at each stage


## Measure the lead time (average time to complete one item, sometimes called "cycle time")

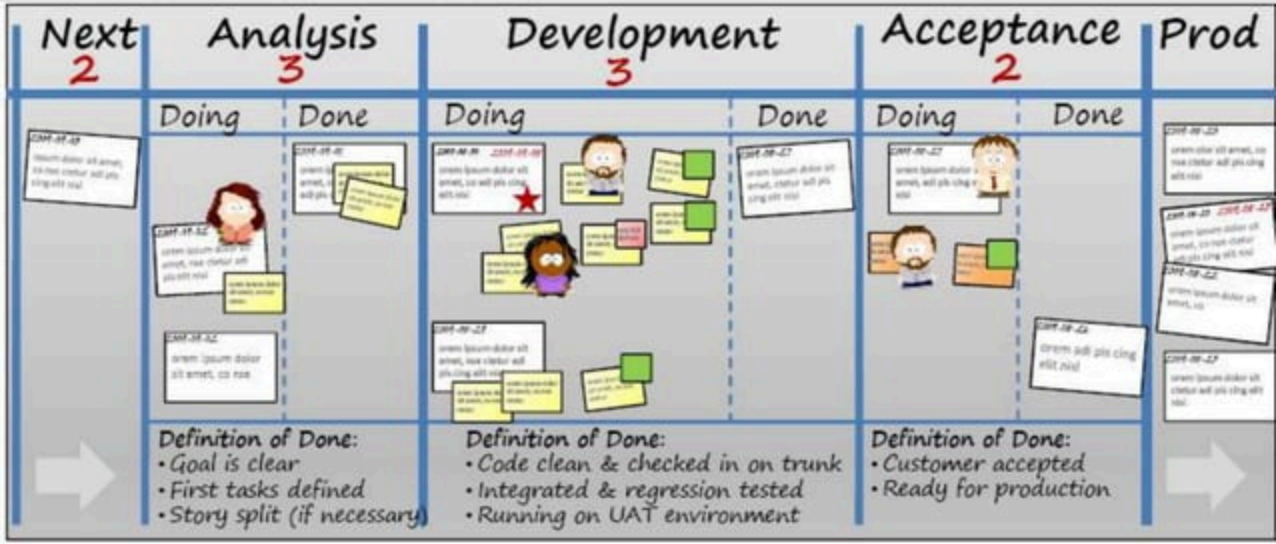
- Optimize the process to make lead time as small and predictable as possible

# Kanban Board Illustration - I



# Kanban Board Illustration - II




## Feature / story

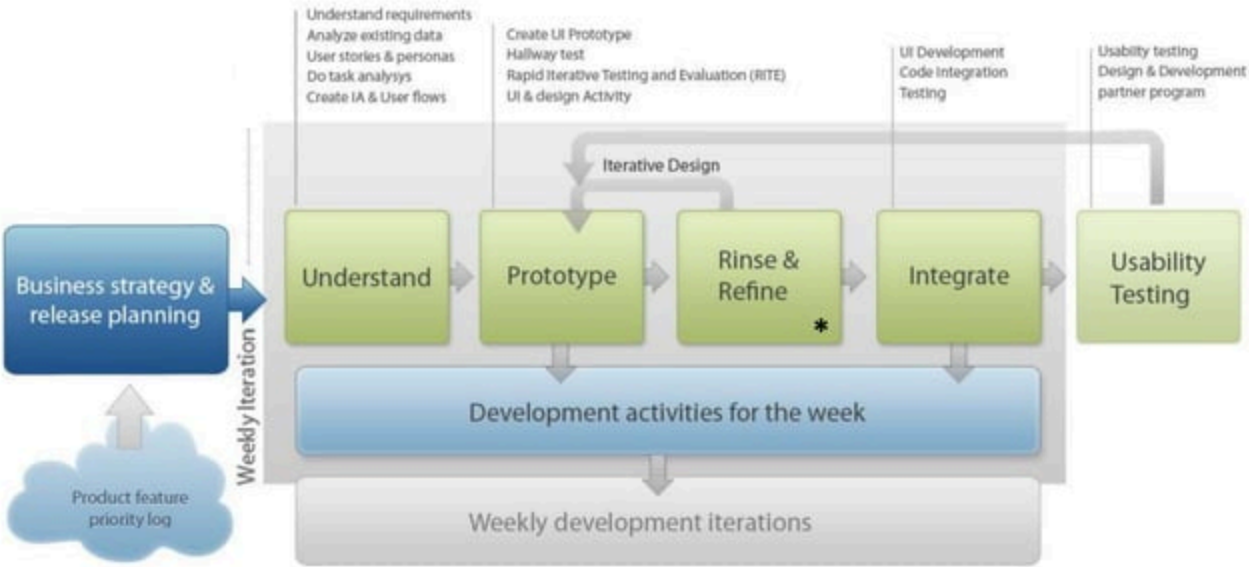

## Task / defect


## What to pull first

1. **Panic** features ★★
   (should be swarmed and kept moving. Interrupt other work and break WIP limits as necessary)
2. **Priority** features ★
3. Hard **deadline** features
   (only if deadline is at risk)
4. **Oldest** features

# UX adopts Agile

# Agile – UX Overlap




* Evaluate internally (sales & marketing) and externally (prospects and clients)

# Resources

- Agile 101
  http://agile101.net/2009/09/08/the-difference-between-waterfall-iterative-waterfall-scrum-and-lean-in-pictures/
- Kanban and Scrum - making the most of both
  http://www.infoq.com/minibooks/kanban-scrum-minibook
- Kanban kick-start example
  http://www.limitedwipsociety.org/tag/kanban-board/

# Thank You