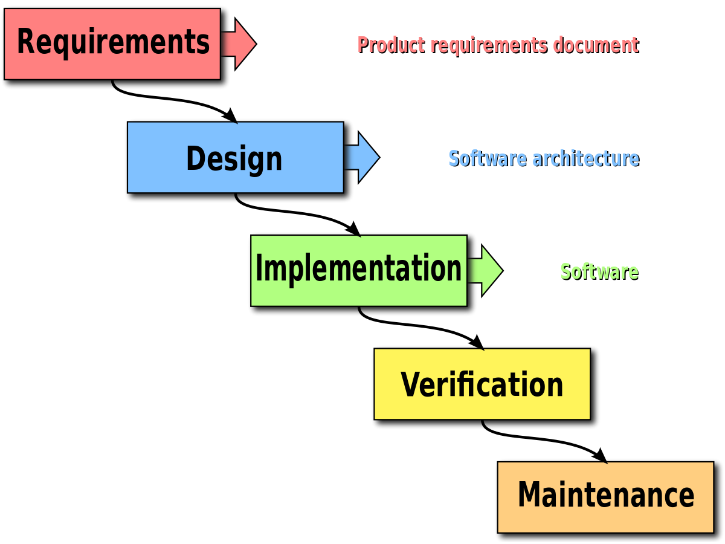
**Learning Agile**

Agile is a set of best practices derived from various process models. The models are waterfall, the prototyping model, the V-shape model and the incremental model.

* 12 principles in Agile, 8 of which are derived from known process models.
* Releases are in continuous increments, as opposed to a waterfall methodology.

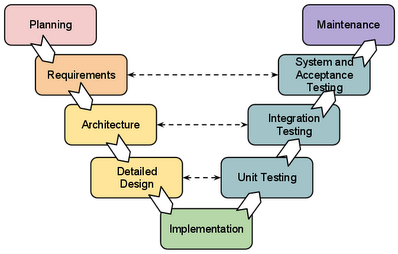
Waterfall disadvantages:

* Late visibility
* Long wait times
* No customer involvement
* No prioritisation of tasks
* Late customer feedback
* Enhanced rework (repeating tasks due to waste)
* No sustainable pace of development
* Not flexible to change
* Limited team ownership
* Measure of progress is process oriented
* Larger scope
* Estimation challenge
* Idle time

**Prototyping Model**

Used when customer isn’t clear on requirements and generates ideas out of working software.

* Rapid Prototype
  + Creates initial prototype for customer
  + Generates further requirements after working with the prototype.
* Throwaway Prototype
  + Create initial prototype for the customer.
  + Dispose the prototype and start afresh.
* **V-Shape Model**
* In waterfall, testing is late in development. The V-shape model plans for testing earlier in the delivery cycle.
* Test Driven Development (TDD)
  + Just have the test cases ready at the start of the development cycle
  + The confirmation criteria of the requirement is defined first form TDD.



**Incremental Model**

It is an iterative waterfall. Every iteration repeats the entire SDLC steps over again. Agile is derived from this model, as an attempt to create a repetitive waterfall.

**Why AGILE?**

Reasons for failing projects and solution:

* Lack of customer involvement
  + AGILE makes the customer a member of the delivery team.
* Poor or vague requirements
  + Requirements are written as acceptance tests just before any test is written. (TDD)
  + A requirement in agile is a story. For each Story there is an acceptance criteria.
* Unrealistic schedules
  + Agile makes estimating and scheduling a collaborative process between customer and the development team.
* Lack of change management in waterfall model
  + Anything can change except for the delivery date.
* Lack of sufficient testing
  + Agile involves continuous build and testing of the code
* Inflexible process
  + Agile integrates retrospection where the current delivery process can undergo continuous improvement.
  + Agile is also referred to as “Process Lite” – lightweight and flexible in terms of delivery processes followed.

**Manifesto for Agile Software development.**

Agile is a lightweight methodology that enables teams to develop software in the face of vague and rapidly changing environment.



Agile plans are shorter (3-4 weeks).

**Agile Principles**

* Customer satisfaction by rapid delivery of useful software.
  + Quickly deliver working software.
* Welcome changing requirements, even late in development.
  + Iterative and incremental development
  + Customer involvement
* Working software is delivered frequently (weeks rather than months)
  + Deliver in short increments
* Working software is the principal measure of progress
  + Not artefacts but functionality delivered
* Sustainable development, able to maintain a constant pace
  + Not back or front loaded systems
* Close, daily cooperation between business people and developers
  + Team collaboration and customer involvement
* Face to face conversation is the best form of communication (co-location)
  + Regular team meetings and collaboration via tools
* Simplicity
  + KISS (keep it simple and sober)
* Continuous attention to technical excellence and good design
  + Product excellence due to small incremental releases and regular customer feedback
* Projects are built around motivated individuals, who should be trusted
  + The team and members are self-motivated
  + Team involvement and hence ownership
* Self-organising teams
  + Due to ownership and roles defines (Scrum master etc)
* Regular adaptation to changing circumstances

**Mastek agile manifesto**

* Customer/business outcomes over internal objectives
  + Deliver to customers objectives rather than the projects objectives
  + Reduce too much focus on internal meetings and milestones
* Continuous improvement over the status quo
  + Improve rather than sticking to the same way of doing things
  + Do not continue what we did in the past if it didn’t work
* Responsiveness over business as usual
  + Reduce waiting between teams for providing services
  + LEAN and Kanban help improve this
* Flexibility/adaptability over following a plan/process
  + Do not blindly continue following a plan
  + Relook at the plan from business objectives perspective
  + Be responsive to change
* Dialogue/teamwork over roles structures
  + Resolve issues yourself with the diret dialogue with that team member rather than escalation
* Proactivity over waiting to be asked
  + Irradiate waiting for receiving services, requested for.

**Terms:**

WOW – Ways of working

Waste – useless work, doesn’t add value (unused code etc.)

TDD - Test Driven Development

SDLC – Software development life cycle

KISS – keep it simple and sober

Key agile terminology

* User story
  + Short description of functionality told from the customers perspective
  + User stories are typically smaller in nature achievable within a defined timeline
  + A very large story is referred to as an epic
  + The business analyst/customer or both write the user story, the business analyst/ customer should be part of daily standup.
  + Attempt to keep user stories independent of each although it would be part of a flow should be able to attempt any part of the plan without affecting the rest of the plan

User stories are made up of 3C’s

* Card
  + Typically documented on an index card and written in just 2-3 lines
  + Format: “As a (role) I want to (objective) so I can (benefit)”
* Conversation- Describes the conversation between various stakeholders
  + Sizing of the user story
    - Discussion of a user story within the team to decide on its complexity
  + Based on its complexity the team sizes the user story
    - This is not effort estimation
    - This is not the time required to solution the user story
  + Sizing is not estimation
* Confirmation
  + Describes the acceptance test criteria for the story
  + Defines when the user story will be complete

Writing a good user story:

* Independent
  + Try to reduce dependency
* Negotiable
  + Story was created after a discussion with the customer
* Valuable
  + What value itll add
* Estimable
  + If you get different estimates even after a team discussion, its not negotiable and hence not estimable
* Small
  + Crisp and to the point
* Testable
  + Always have some confirmation mechanism

Product backlog

-

Definition of done

* Decides whether a user story is done/complete
  + Tested
  + Bug fixed
  + Delivered deploued
* May vary for each team
* Decided post discussion within team
* Also called sprint goal

Time-boxed

* planning technique used in agile projects where the schedule is divided into a number of separate time periods
* Each time period is of a fixed duration is called a “sprint”
* A sprint ends once the time period elapses (typically 2-4 weeks)
* Large projects can have multiple sprints for multiple modules
* Teamsize should be 5 to 9

Sprint

* Sprints are typically of the same length, end date is fixed, no change.
* Initially for the first sprint we may go with experience to get the duration right, after which the duration should stabilise.
* We have the option of modifying the length in the next sprint onwards, but it should be constant then on.
* Any work remaining in the sprint has to stop when the sprint duration completes
* The sprint has either succeeded/completed or failed.
* New requests cannot be accommodated in between a sprint and should be accommodated in the next sprint.
* If a sprint is not delivering value to the customer, the team should stop the sprint and discuss with the customer and find a way to deliver value to the customer on sprint completion.

Velocity

* It’s the amount of value delivered in each iteration, measured in story points day or hours.
* Only completed stories are counted for calculating velocity.
* Velocity is useful for planning for features and releases.
* It is important for the sprint duration to be constant else it would impact the velocity.
* The number of resources per sprint shouldn’t vary largely.
* Plan sprint wise and not for the entire product.

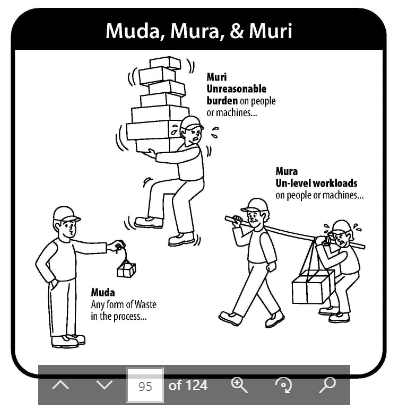
Scope creep

* More user stories added in between by the customer
* Changes and defects may also get added.

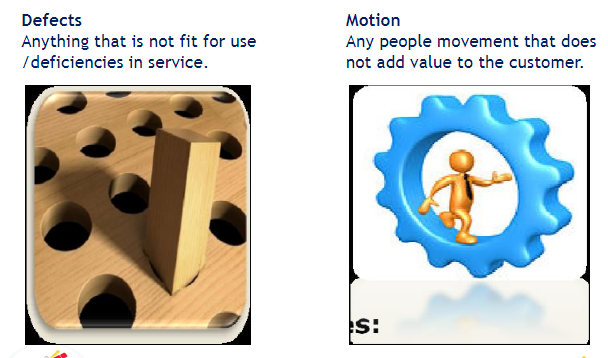
LEAN

Lean is a philosophy that focuses no continuous elimination of waste, inconsistency and unreasonableness within the organisation, using a set of tools and guidelines.

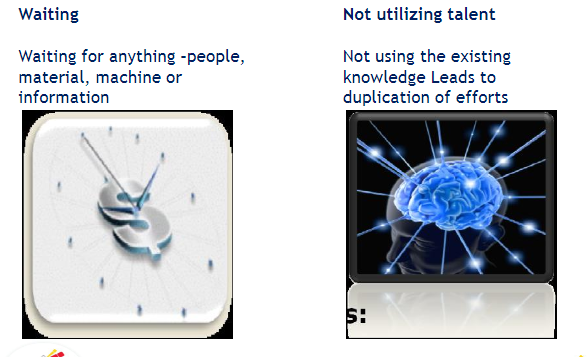
Understand the logic and rationale behind every process step rather than just accepting it as it comes.

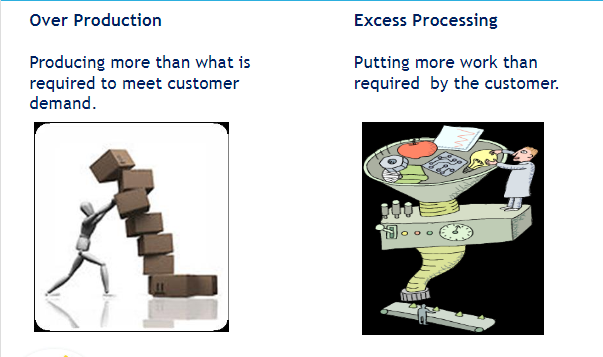












Examples of DOWNTIME

1. Missed schedules - defects
2. Changing from one task to another or looking for information etc. - motion
3. Developers waiting for requirements - waiting
4. Endless refinements to reports – excess processing
5. Multiple forms with same information produced – over production
6. Cross skilled resources not used – not utilising talent
7. Documents moving from one department to another for approval - Transportation
8. Software coded but not tested, lying in the system - Inventory

Value Added Activities

Lean focuses on improving the proportion of value added activities and reducing the non-value added activities.

* Value Added (VA)
  + Product or service is transformed into a state required by the customer
  + The customer is willing to pay for it
* Non Value added (NVA)
  + Activities which consume resources but don’t create value.
  + Customer is not willing to pay for these.
* Non value added but needed(NVAN)
  + Activities causing no value add but cannot be eliminated based on current technology or thinking