3.Scrum

3.1 Introduction

Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.

the business sets the priorities.

every 2-4 weeks runnable and updated sw.

relay race: instead of focus on the reliability and flexibility of the sw, today reqs focus on speed of release.

origin in 90s but founded in 2002.

Scrum is used even in it companies and in not it companies.

3.2 Characteristics

team self-organize to determine the best way to deliver.

2-4wks sprints.

reqs are listed (product backlog)

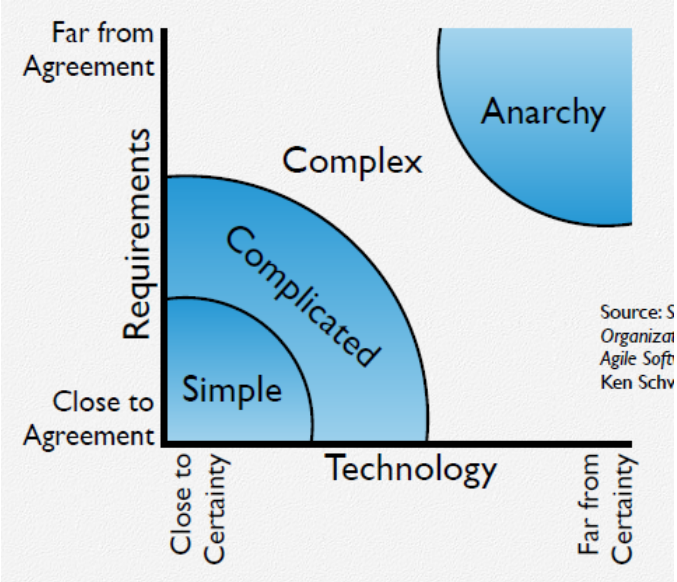
no specific practises

Scrum is a Agile process

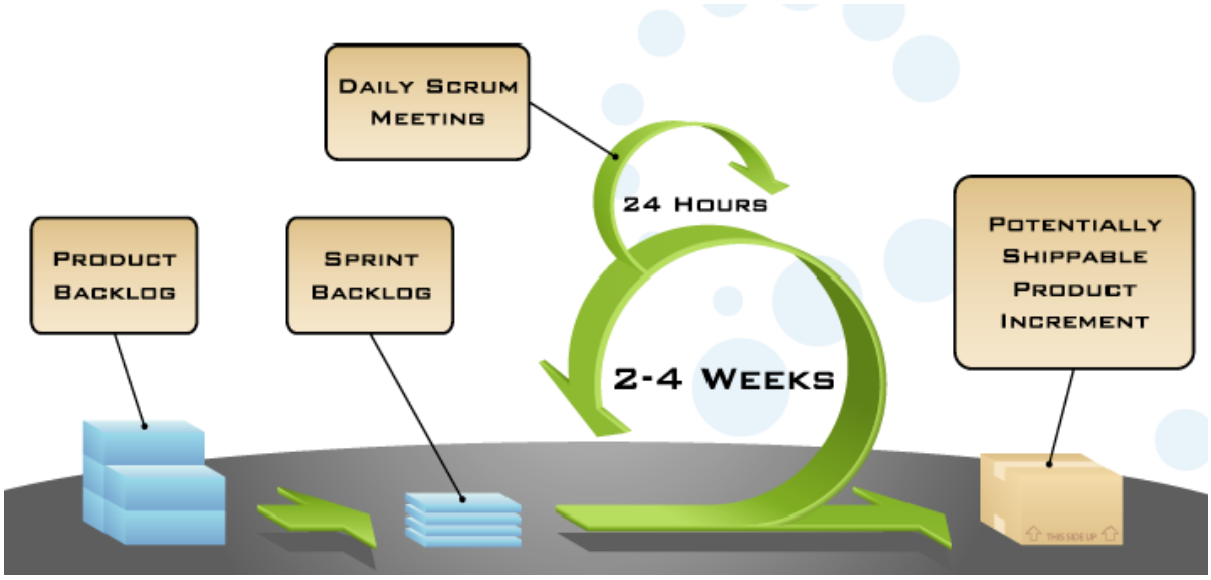
- Agile Manifesto

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

- Project noise level



3.3 Phases



- Features should be ordered and prioritized, so in any time you know which feature to realize first.
- Sprint backlog - define at a more fine-grained level what are the piece that should be done in order to realize the sprint.
- Sprint - duration of one iteration of the process.
- At the end of the sprint you have a potential shippable product.
- During the sprint, the whole team have to have a daily scrum meeting.

3.4 Sprints

Scrum project are divided in "sprints" of 2-4 weeks

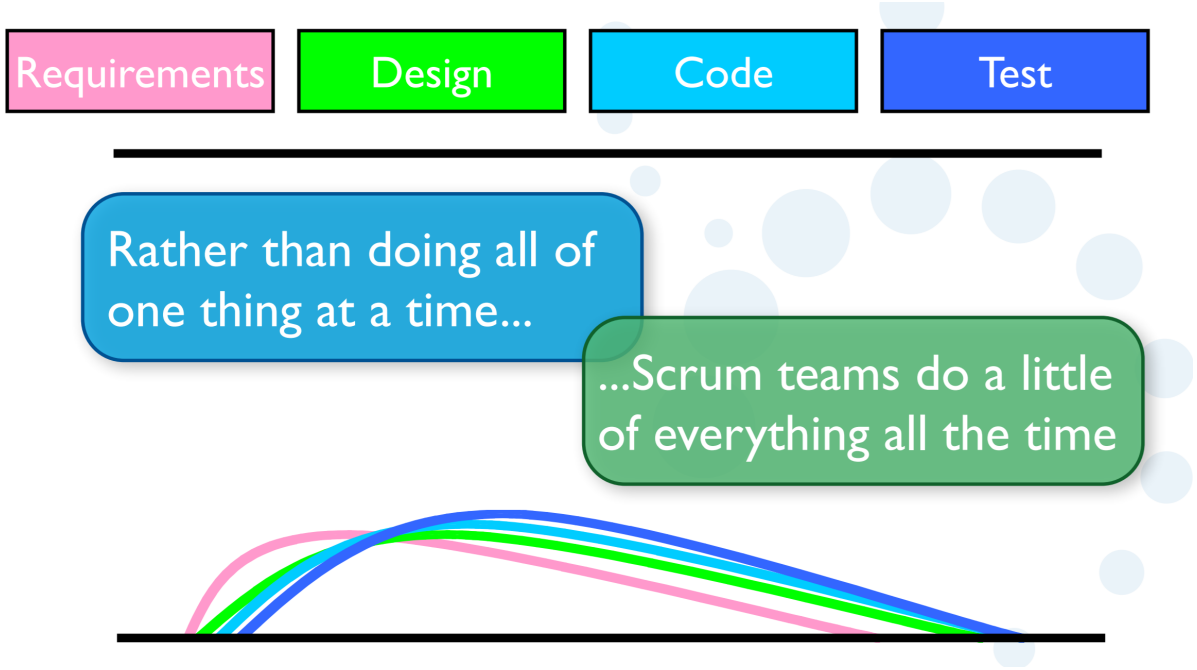
duration is always the same so the team have freedom.

it's important that the product is designed, coded tested, during the sprint →sw should run properly.

No changes during sprint. The changes will be done in the next sprint and riprioritized by the owner.

- sequential vs. overlapping dev

  In scrum you don't have to do all design, code or test at a time, but do a little of everything for every sprint



3.5 Scrum Framework

3.5.1 Roles

- **Product owner**

  The guy that is in charge of manage (the guy that put the money cit.)

  decide release date

  responsible for the profitability (ROI).

  define priorities for the next sprint

  accept or reject work

- **ScrumMaster**

  manage the project, responsible for the scrum practices, solve issues, shield the team from external interferences.

- **Team**

  5-9 people, cross-functional, full-time, self-organizing, change only between sprints (not during).

3.5.2 Activities(ceremonies)

- **Sprint planning**

  box that take some input: team capacity, estimation work,

then: sprint planning meeting:

1) sprint prioritization(analyze and evaluate prod, select goal) → sprint goal

2) sprint planning(how to achieve goal, create sprint backlog from user stories, features, estimate sp-backlog in hours) → sprint backlog

- **The daily scrum**

  daily,15 mins, stand-up (at the coffee machine), all the world invited, not for problem solving, only the whole team can talk.

- **Sprint review**

  informal, no slides, 2-hour prep, the form of a demo of new features.

- **Sprint retrospective**

  what is and is not working, 15-30 mins, after every sprint, the whole team decide if start/stop/continue

3.5.3 Artifacts

- **Product backlog**: list of reqs(with time estimation), prioritized by owner and reprioritized at the start of every sprint.
- **Sprint backlog**: goal → the objective of the sprint, what the work will be focused on.

  manage product backlog → update and manage time remaining to achieve the goal.

- **Burndown charts**: charts that show how the work is going.

  display what it has be done and what not(overall progress).

3.6 Scalability

7+-2 people

Factors in scaling:

Duration

Team size (you can do scrum of scrum)

Team dispersion

Type of app.

3.7 User stories

3.7.1 User stories

Instead of Use Cases, Agile project owners do "user stories"
Who (user role) – Is this a customer, employee, admin, etc.?
What (goal) – What functionality must be achieved/developed?
Why (reason) – Why does user want to accomplish this goal?

As a [user role], I want to [goal], so I can [reason].

Example:
"As a user, I want to log in, so I can access subscriber content."

story points: Rating of effort needed to implement this story
common scales: 1-10, shirt sizes (XS, S, M, L, XL), etc.

3.7.2 Behavior-Driven Design (BDD)

Asks before and during dev (v&v in a better way).

reqs as user stories

concentrate before on the behvior and after on the implementation

the complement is the tdd (Test Driven Development).

3.7.3 Measuring Productivity

Velocity: avg number of point/week.

Fibonacci scale is used for decide point for each feature(1,2,3,5,8).

The team votes to decide the point of each feature.

stories should not relate with layers (front-end, back-end, js) but just features.

3.7.4 Smart stories:

specific, measurable(Given/When/Then), achievable(finishable in <1 iteration), relevant(should have a business value, 5 whys), timeboxed(stop when exceed time budget).

3.7.5 Lo-Fi UI Sketches and storyboards

Pay attention to envision the UI you want because a user story has a stakeholder behind.

Lo-fi ui → pen & paper

Easier than product html

Prototypes are expensive in term of time

3.7.6 Storyboards

Need to show how UI changes based on user actions
Like scenes in a movie,