Trivia

- Where do we find earliest references to the following sequence of numbers [1, 2, 3, 5, 8, 13, 21, …]?
    - in works of Leonardo Da Vinci on golden ratio ~ 14th century
    - in works of Leonardo Fibonacci on a population of rabbits ~ 13th century
    - in works of Hindu scholars ~ 200 BC
    - in works of Soviet Union mathematicians ~ 500 BC
- Which of the following is incorrect interpretation of the word SCRUM?
    - an acronym for [**S**]oftware [**CR**]eation [**U**]nified [**M**]ethodology
    - a rugby term for a way of starting play where players from each team come together and try to get control of the ball by pushing against each other and using their feet when the ball is thrown in between them
    - a large group of people who are close together in one place
- Which of the following certifications can't be used in the context of professional malpractice claim?
    - CNA
    - CLA
    - CSPO

Background

- "Agile" is the unifying term for number of approaches/methodologies to software development process. It appeared post factum, after number of these more concrete methods evolved to took roots:
    - XP
    - TDD
    - Scrum (which is simply one of them)
    - ….
- Many popular methods we use in SD today come from other industries - mainly manufacturing. Many of them in some form or shape could be traced to Japanese manufacturing during post-war revival
- There are other methodologies in use in SD that do not fall under Agile umbrella, f.e. Kaizen or Waterfall :-)
- First references to WF ~ 1970, Agile ~ 2001
- Agile is usually thought of as anti-Waterfall, but under some angle Agile can be seen as a sequence or iterations of small "Waterfall"s (this may help during estimation). Agile is "iterative" process whereas Waterfall is "sequential"
- Agile is not a silver bullet that solves famous iron triangle. Even with most elaborate Scrum process in place the balance between three vertices is something every organization has to struggle with

- Scope
- Date
- Cost

- Agile Manifesto vs Software Craftsmanship Manifesto

  http://www.agilemanifesto.org/
  http://manifesto.softwarecraftsmanship.org/

Scrum

- Process organized around teams (7+/- 2) working together in Sprints to deliver working code at Sprint Review
  - Based on this “official” definition smaller teams and ultimately single-person projects (f.e. in case of independent contractor) may not benefit from Scrum process
  - In this definition team size includes PO and SM (a detail, that formally differentiates “The Scrum Team” from “The Team”)

- 3 Roles
  - Product Owner (aka, Product Sponsor)
    - holds the vision
    - represents customers
    - creates acceptance criterias
    - always available for Q&A
    - ...
  - Team Member
    - responsible for delivering user stories
    - self-organizes to do achieve best results in doing so
    - owns estimation
    - includes QA!
    - ...
  - Scrum Master
    - expert, guardian and facilitator of the process
    - not team’s boss position, but peer position set apart by additional responsibilities
    - first “go-to” person for any team member in case of road blocks during Sprint
    - this role could be shared from sprint to sprint, especially in teams with “contributing scrum master”
    - …

- 3 Ceremonies (plus 1)
  - Sprint Planning
    - PO identifies number of stories to be considered for a sprint along with their priorities
    - (part 1) in discussion, team commits to selected number of stories (that should be estimated by that point ideally)
    - (part 2) team breaks down stories into tasks targeting at least 60% of accuracy
    - ...

    - Daily Scrum (aka Standup)
        - what accomplished yesterday
        - plan for today
        - blockers

    - Sprint Review (Sprint Demo)
        - show & tell about anything interesting in the context of delivered stories: working UI feature, Server feature if possible to demo, design doc, refactored code, etc.
        - strict time constraints, tailored to the audience in the room

    - *- Sprint Retrospective -*

- 3 Artifacts (plus 1)
    - Product Backlog
        - owned by Product Owner
        - has technical stories vs user stories (creative tension similar to AM vs SCM)
        - constantly being groomed and prioritized
        - stories on the top should, ideally be pre-estimated by the team
        - ...

    - Sprint Backlog
        - fixed list of stories identified in sprint planning meeting - “to-do” list for the current sprint
        - while list of stories is fixed, list of underlying tasks will change
        - strictly speaking, PO may not change list of stories in the sprint backlog once sprint has started
        - …

    - Burn Down Chart
        -

    - *- Done Criteria -*
        - *different from acceptance criteria defined by PO in a way that in addition it may also include selected and agreed upon “internal” deliverables… As an example:*
            - *APIs or schema changes published on Wiki*
            - *applicable unit tests*
            - *applicable performance metrics*
            - *records of code review*
            - *etc ...*

Sprint Artifacts could exist in any form, from sticky notes to spreadsheets. Our Jira is great way to keep it integrated with rest of the process and tailored to the needs of the team. We have 3 lanes currently (“Not started”, “In Progress”, “Done”). Some team add lane “Blocked” for example.

Story points

Overall Scrum process could be thought of as a way to mitigate uncertainty. To deal with uncertainty we have to estimate and predict our efforts.

It is known that as time of delivery extends in the future, precision of time or cost estimate declines - not linearly but close to exponentially. Fibonacci numbers reflect this phenomena. When used for Scrum, sequence stops at around 100. This signifies that story of larger size, can not be estimated in a way that will be useful for planning, and that we need to break down this story into smaller one.

People can inherently compare things, while not always be able to measure them directly. We can differentiate clearly between 1 story building and 3 story building. Most people can evaluate (without counting!) quite precisely how tall given building is up to 5 stories. When it comes to taller buildings precision of direct evaluation or estimate drops down. At the same time, knowing size of one building we can quite precisely evaluate size of another, again up to certain limits. So, when it comes to use of F.N. for estimation, we say that we know quite precisely difference between effort required by a task estimated as 2 and a task estimated as 3. But distinction between tasks evaluated as 13, 14, 15 or even 19 is blurred (in terms of how precise it is). Only at 21 estimated points we can “detect” definite change of precision of estimation. And so forth. (But we should remember that use of F.N. is just a method, helpful convention and can’t be thought as something “absolute”... At the same time things like “pound”, “meter”, “hour” are also just a useful conventions)

When developers are asked to estimate task in hours - value will vary drastically from person to person. Fact that person A performed task T1 in 2 hour, while person B performed the same task in 1 day, does not say in itself anything about task and its “objective” complexity. Hours are subjective, and as such are hard to use of sustainable long term planning.

We use story points, then, as units reflecting “objective” value of complexity of a task as it exists in the context of a given team. It makes sense only as a relative value with some pre-existing empirical point of reference (based on previous efforts). Different teams will assign different number to task complexity - it could be driven by team’s composition, specifics of the project and environment. But once defined, as time goes by, similar tasks will receive more or less similar assignment of complexity.

Provided that each member of a team does their Best in a Sustainable manner within a period of time called Sprint, a team will accomplish by the end of Sprint a certain amount of work of a good quality. “Sustainable” - means that amount of work produced allowed engineers to feel “balanced and in the zone” and that this amount can be relied one periodically and for long time. “Best” - means that this work has highest quality possible and done in the most efficient manner by each member of a team.

At the end, it is not raw time estimate that matters but this predicted with empirically known certainty amount of work that team is known be able to deliver in a “sane”, sustainable and qualitative way. This give more planning power and certainty and flexibility to PO as opposite to a direct answer to the question “How long will it take?”

There are always, special times and circumstances that drive important delivery dates (trade shows, holidays, product releases by competitors, etc…) - and team will go out if its way to deliver according to these special needs. But if this is not an exception but a rule - the process will not be sustainable, and sooner or later something will give.

Breaking stories into tasks - dealing with “writer’s block”

- It is often seen as something “unnatural” or “hard to do” to break Tasks down to concrete Subtasks - especially for new members of Scrum teams

- Sometimes we all feel that there must be more granular tasks to the given story, but we can't pinpoint them and identify formally
- Following could be helpful in these cases:
    - Remember that stories often composed of tasks that "resonate" with overall development environment. As such thinking about life cycle of the feature or deliverable can help identify smaller pieces involved
    - Often potential implementation details (as taboo as it sounds) can lead to good sizable generic tasks - f.e. Story=new Java Package; Task 1=new Service/Manager classes; Task 2=new Dao classes; Task 3=new Database objects
    - Good ideas can be found among sub-titles on the auto-generated Server Design Spec Wiki (:-))
    - Look among closed stories on similar subjects