Agile Software Development and Scrum

* Project
  + maintain a lot of documentation
  + use software dev process
  + contributions
  + commits
  + code reviews
  + sprint planning
* Agile Manifesto
  + created an agile alliance
  + core principals (4)

|  |  |
| --- | --- |
| individuals and interactions over process and tools | * + - * people make biggest impact on succces         + process and environment help, but will not create success       * strong individuals not enough without good team interaction         + individuals may be stronger based on their ability to work on a team       * tools can help, but bigger and better tools can hinder more than help         + simpler tools can be better |
| working software over comprehensive documentation | * + - * documentation important, but too much is worse than too little         + long time to produce, keep in sync with code         + keep documents short and silent       * focus effort on producing code, not descriptions of it         + code should document itself         + knowledge of code kept within the team       * produce no document unless its need is immediate and significant |
| customer collaboration over contract negotiation | * + - * not reasonable to specify what's needed and then have no more contact until final product delivered       * get regular customer feedback       * use contracts to specify customer interaction rather than requirements, schedule, and cost |
| responding to change over following a plan | * + - * environment, requirements, and estimates of work required will change over course of large project       * planning out a whole project doesn't hold up         + changes in shape, not just in time       * keep planning realistic         + know tasks for next couple of weeks         + rough idea of requirements to work on next few months         + vague sense of what needs to be done over year |

* 12 Principles behind the Agile Manifesto
  + Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
  + Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
  + Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
  + Business people and developers must work together daily throughout the project
  + Build projects around motivated individuals. Give them the environment and support they're need, and trust them to get the job done.
  + The most efficient and effective method of conveying information to and within dev team is face-to-face conversation
  + Working software is the primary measure of progress.
  + Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
  + Continuous attention to technical excellence and good design enhances agility.
  + Simplicity--the art of maximizing the amount of work not done--is essential (focus on core and essential)
  + the best architectures, requirements, and designs emerge from self-organizing team
  + At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

* based on these principles, many lifecycle models were developed (35:00)
* Diagram

  Description automatically generated

* XP : Extreme Project Programming (most popular)
  + more heavy-weight
  + code review
* Scrum (most popular now)
  + 9 different rules
  + does not mandate peer programming
* Kanban (lightweight)

* What is Scrum
  + team work matters

|  |  |
| --- | --- |
| all members are working together for a single goal | deliver working software |

* + Ken Sharper (creator of scrum)
  + Graphical user interface

    Description automatically generated with medium confidence
  + team must be cross-functional
    - design
    - testing
    - development
    - release engineering
  + team must be self-organizing
    - team must figure out best way to develop something

|  |  |  |
| --- | --- | --- |
| 3 types of role | * + - 1. product owner       2. scrum master       3. scrum team |  |
| scrum teams | * + - 1. software | * + - 1. split into small features/components       2. prioritize / sorted list          * product background (scrum)          * software requirement specification (waterfall) |
| product owner | * + - 1. prioritize       2. technical & business knowledge       3. good communication skills essential       4. liason btw team and client | * + - 1. define the features       2. make scope vs schedule decisions       3. responsible for achieving financial goals of the project       4. prioritize the product backlog       5. adjust the feature and priority every sprint, as needed       6. accept or reject work results |
| scrum master | * + - 1. facilitator       2. team reports up to       3. schedules meetings          * weekly best       4. first to look into solving problems       5. maintain       6. manage project       7. resolve team conflicts | * + - 1. responsible for enacting Scum values and practices       2. removes impediments       3. Coaches the team to their best possible performance          * Helps improve team productivity in any way possible       4. Enable close cooperation across all roles and functions       5. Shield the team from external interference       6. RULES = SCRUM PRINCIPLES          * ensures team is following |
| the team | * + - 1. 5 to 9 people | * + - 1. cross-functional          * programmers, testers, user experience designers, etc.       2. members should be full-time          * may be exceptions (e.g., database administrator)       3. teams are self-organizing          * ideally, no titles but rarely a possibility       4. membership should change only between sprints |
| sprint backlog | * + - 1. release cycle | * + - 1. product backlog          * small product tasks required to complete feature   prioritized by product owner   * + - 1. determine num hours due |
| scrum meeting | * + - 1. daily | * + - 1. team standup          * what you did          * what you are planning to do          * is there any problem in your work |
| release | * + - 1. potentially shippable product increment |  |

* Waterfall
  + like a relay race
    - 4 runners, passing baton, the fastest runs 100 meters…
      1. they working as a team, but not working at same time
  + team analyze requirements, finalize it, send to design team
  + design team will create design off of requirements
  + send to code team (39:00)

* Sequential vs. Overlap

|  |  |  |  |
| --- | --- | --- | --- |
| Requirements | Design | Code | Test |

* + - Rather than doing all of one thing at a time…
    - … Scrum teams do a little of everything all the time
* Sprints
  + Scrum projects make progess in a series of "sprints"
  + Typical duration is 2-4 weeks or a calendar month at most (ideal)
    - every sprint has some overhead
      * too short reduces productivity
      * too long creates problems
      * procrastination
    - timeline depends on client
  + A constant duration leads to a better rhythm
  + Product is designed, coded, and tested during the sprint
* Scrum Framework
  + Roles
    - Product Owner
    - Scrum Master
    - Team
  + Artifacts
    - Product Backlog
    - Sprint Backlog
    - Burndown charts
  + Ceremonies
    - Sprint planning
    - Sprint review
    - Sprint retrospective
    - Daily scum meeting
* Scrum Roles
  + Choose roles for project team

Scrum Ceremonies

* Sprint Planning Meeteing
  1. team capacity (can change/recalculate ea. sprint)
     1. how many features we can complete during sprint
     2. how many stories can we complete during spring
        1. team velocity
  2. product backlog
  3. business conditions
  4. current product
  5. technology