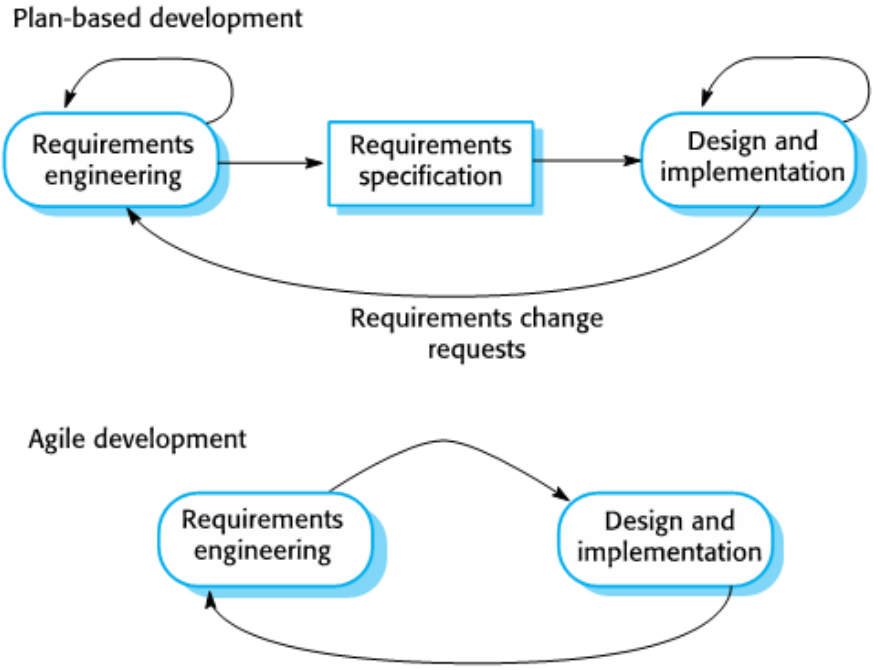
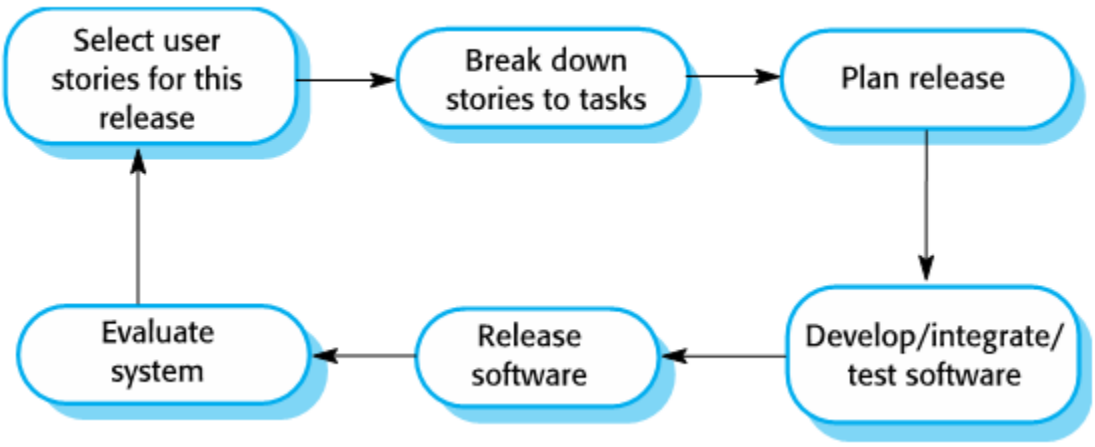
Software engineering notes:

- Rapid software development is essential nowadays because:
  - business operates fast
  - software should evolve quickly
- Plan driven approaches do not allow for such. Hence agile dev used



- Plan driven:
  - separate development stages with output planned in advance
- Agile:
  - interleaved from specification, design and testing and outputs are decided through negotiation during software development
- If detailed specification required then use plan driven. If incremental strategy to deliver and getting feedbacks then use agile.
- Agile features:
  - minimal documentation, advanced tools for development
  - frequent delivery of new versions
  - stakeholders involved in version specification and evaluation

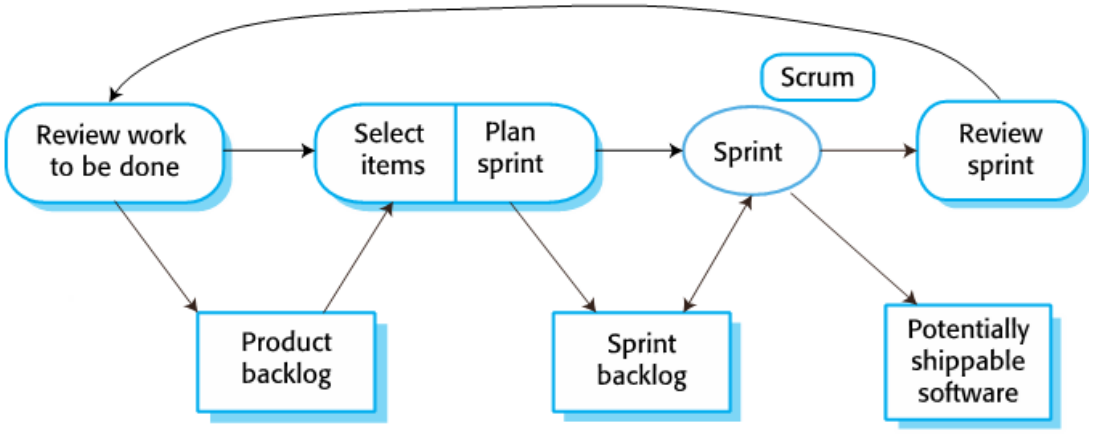
- Agile aims to reduce overheads and be responsive to changing requirements
- Manifesto:
  - individuals rather processes/tools
  - working software rather than documentation
  - customer collaboration rather than contracts
  - responsive rather than plan based
- Principles of agile:
  - customers are involved to prioritize and specify requirements at each iteration
  - responsive to changes
  - maintain simplicity of the system
  - let developers work at their own pace and assign them per their own skillset
  - good team communication
- Applicable where:
  - for small, medium sized products. Nowadays for large products
  - less external hinderances and commitment of customer to be involved
- Development Techniques:
  - Extreme Programming:



    - extreme approach of iterative development, increments delivered every two weeks, builds are created and tested severally. However, not easy to integrate hence management practice required

- Practices:
  - simple design to meet requirements only
  - test-driven development
  - code refactoring
  - user story for specification
  - pair programming
  - no one has ownership over the code, all constitute same rights
  - Continuous integration of code and a sustainable pace is maintained
- XP + Agile common practices:
  - small, frequent system releases
  - pair programming, collective ownership, no long hours
  - constant refactoring
- Description of some XP practices:
  - User stories for specification:
    - user stories of requirements made and written on cards
    - customer being a part of XP team chooses next released based on personal priority
  - Refactoring:
    - constant code improvement
    - it is wise to spend time in anticipating changes to reduce future cycles of work
    - Examples:
      - removing duplications
      - following name convention of attributes
      - replacing inline code with calls
  - Pair Programming:
    - egoless programming
    - involves programmers working in pairs to develop code
    - pairs are made dynamically
    - Advantages:
      - develops a common ownership
      - serves a constant review process

        - encourages refactoring
        - avoid risks if teams are changed
    - Some XP testing features:
        - Test-First Development:
            - Program is tested once every change is made
            - Tests are written before actual code in the form of programs to check for correction and errors.
            - Frameworks as Junit
            - Some problems:
                - test may be incomplete
                - difficult to write incrementally
                - may not cover all test cases.
        - Customer involvement:
            - customers writes/helps in developing acceptance tests to ensure customer needs
            - However, such a mainstream approach is not taken mostly by customers
        - Test automation:
            - Tests are written as executable components before implementation
            - Helps in verifying credibility of newly written code
- Project management:
    - management is required for on-time delivery within the planned budget. Plan driven involves answering what deliverables, when and who will work on the project.
    - Agile follows a different approach.
    - Scrum:



- agile method that focuses on iterative development
- Terms:
  - Sprint: a development iteration
  - Velocity: Amount of product backlog covered in single sprint
- Scum team has:
  - Scrum master:

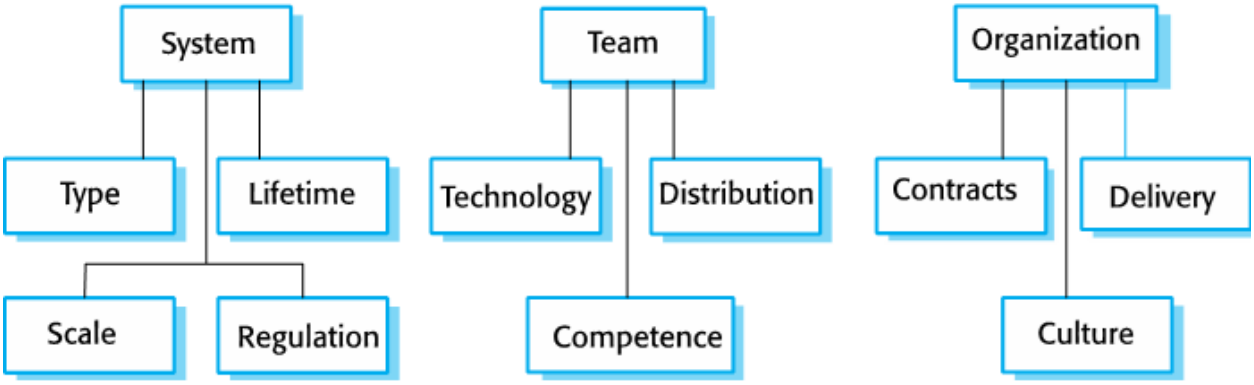
    To ensure scrum process is followed properly and interface with the rest of the company
  - Product Owner:

    Product manager or customer; that identifies product features, requirements and reviews the product backlog to ensure correctness
  - 7 Developers:

    For developing software and other documents
- Artifacts:
  - Product Backlog:
    - ‘to-do’ list. Software requirements, user stories, user documentations etc.
  - Release Backlog:
    - select items from product backlog for first release; this subset is release backlog

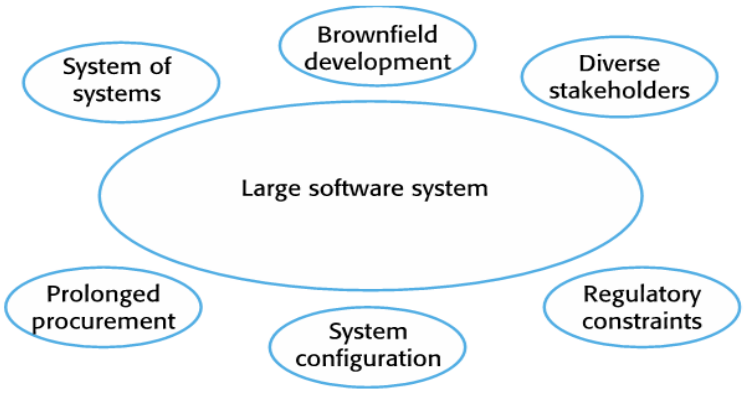
        - Sprint Backlog:
            - User stories selected by the developers to implement in the current sprint cycle.
        - If some changes are prioritized, these are pushed onto product backlog again with priority
- Phases:
    - Sprint planning meeting:
        - 2-4 hours
        - Team decide on the sprint goal
        - Sprint backlog made. Velocity, capacity also decided
    - Sprint:
        - 2-4 weeks
        - Teams work to finish an increment independently; are isolated
    - Daily Scrum:
        - brief meetings to give daily status to product owner
        - to see if every one is on the same page; what was done and what is to be done?
    - Sprint review meeting
        - 1 hr
        - demo the sprint release/increment to gain feedback
        - Product owner may re-work product backlog based on the current sprint
    - Sprint retrospective
        - Inter sprint meetings
        - decide on how to improve the next sprints, review mistakes etc.
- Benefits:
    - improved team communication
    - makes things easy and manageable
    - insight of progress of the project
    - on-time delivery of increments and feedback
    - trust of customer gained with successive iterations

    - Distributed Scrum:
        - remote teams; Individuals working in the same team but located in different locations.
        - Scrum masters should be located with the developers
        - Product owner should visit developers to establish trust
        - Video conferencing and other real time tools used
        - Continuous integration so that insight of each sprint is known
- Scaling Agile:
    - Scaling: changing agile to cope up with multiple distributed teams working on a large project
    - Scaling up: developing large projects that cannot be made by small teams
    - Scaling out: introduction of agile methods across a large organization
    - Must maintain agile principles
- Practical problems:
    - incompatible with contract definition:
        - contracts are required in order to gain customer trust of delivering correct product. However agile only follows it minorly
    - good for software creation rather than maintenance:
        - Problems:
            - lack of documentation
            - keeping customers involved
            - maintaining the same development team which is not always the case hence misunderstandings
    - not ideal for large distributed teams
    - Some plan-based factors:



- System:
  - Scale: agile good for small co-located projects
  - Type: Detailed analysis demand more documentation
  - Lifetime: greater lifetime, more documentation needed
  - Regulation: if external regulation, then more documentation
- Team:
  - Technology: Betters tools needed if no documentation
  - Distribution: Distributed team require documentation
  - Competence: agile needs high skill rather than plan-based workers
- Organization:
    - Are contracts needed for system specification?
    - Can feedback be gained for system increments?
    - Can agile fit to the documentation culture?

- Agile for large systems:



- Problems of using agile for large systems:
  - less flexibility for incremental development
  - not possible to focus only on the code of the system as cross-team communication mechanisms have to be designed too
  - Continuous integration is practically impossible
  - cannot be a single product owner
- Some use Multi-team Scrum:
  - Role replication: separate product owner and scrum master
  - Product architects: separate architects that all collaborate to form the system architecture
  - Release alignment: release dates are aligned
  - Scrum of Scrums: meetings involving each team representative
- Project managers may be reluctant to accept this
- some cultural resistance; organization don’t shift because of having conventional engineering processes