Project management - methodologies

Jana Kostičová, 13.12.2023

Methodologies - project vs. process

Project management (PM) methodologies vs. SDLC models & methodologies

- Closely interrelated - the chosen project methodology affects the SDLC model / methodology and vice versa
- The goal is to create synergy between them to ensure a cohesive and effective approach to software development

Generally it holds

- Sequential PM methodologies align well with sequential SDLC models
- Iterative-incremental PM methodologies align well with iterative-incremental SDLC models

Methodologies

Traditional / sequential methodologies

- Waterfall
- Critical Path Method (CPM)

PMBOK (by Project Management Institute)

- A set of standards

Agile approaches

- Pure Agile
- Scrum
- Kanban
- Extreme Programming (XP)
- Feature-driven development
- SAFe (Scaled Agile Framework)
- …

Process oriented

- Lean management
- Six sigma
- …

Other

- Prince2 (Projects IN Controlled Environments)
- …

Waterfall & iterative-incremental approaches

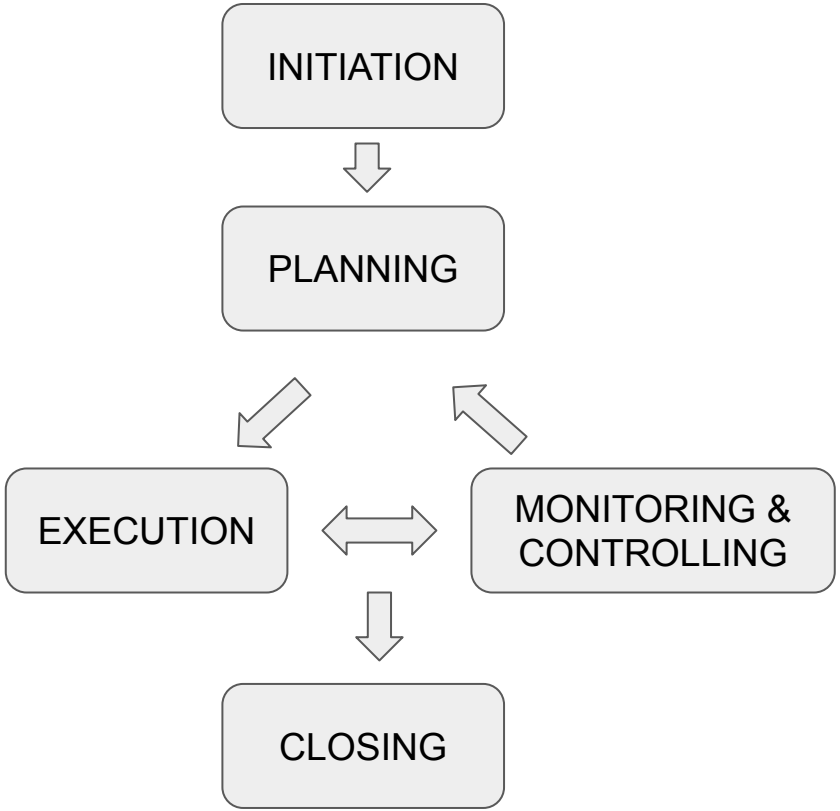
Similar principles to SDLC

Waterfall

- Project phases follow one another sequentially (note that “execution” and “monitoring / controlling” are inherently parallel)
- Planning is usually done only once, although the project plan may be updated during subsequent phases

Iterative-incremental approaches

- Breaking down a large project into smaller, manageable iterations with incremental releases
- Basic approach - “small” interconnected waterfalls
- Other approaches - e.g. agile methodologies



Critical path method - CPM

= is a project management technique used to plan and manage complex projects (also Critical Path Analysis - CPA)

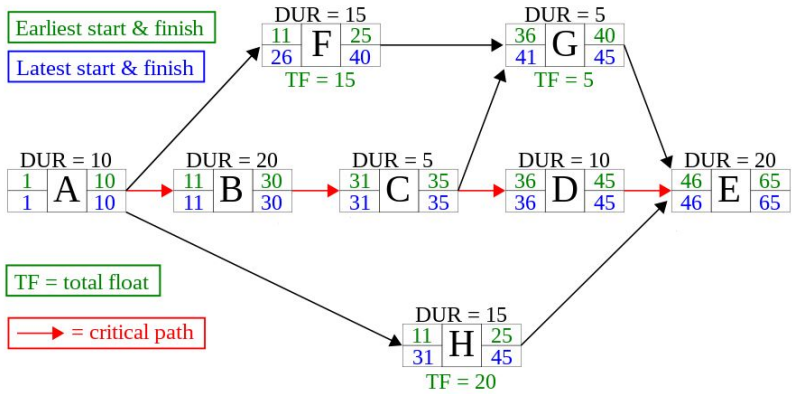
Critical path (CP): The longest path through the project, any delay on this path would directly impact the project's completion date. CP determines the shortest time possible to complete the project.

Critical activities: Activities on the critical path

Total float (TF): The amount of time an activity can be delayed without delaying the project.

- Critical activities have zero TF
- Non-critical path activities have positive TF, indicating flexibility in their scheduling

CPM is frequently used in the Waterfall model, however it is generally not restricted to this approach

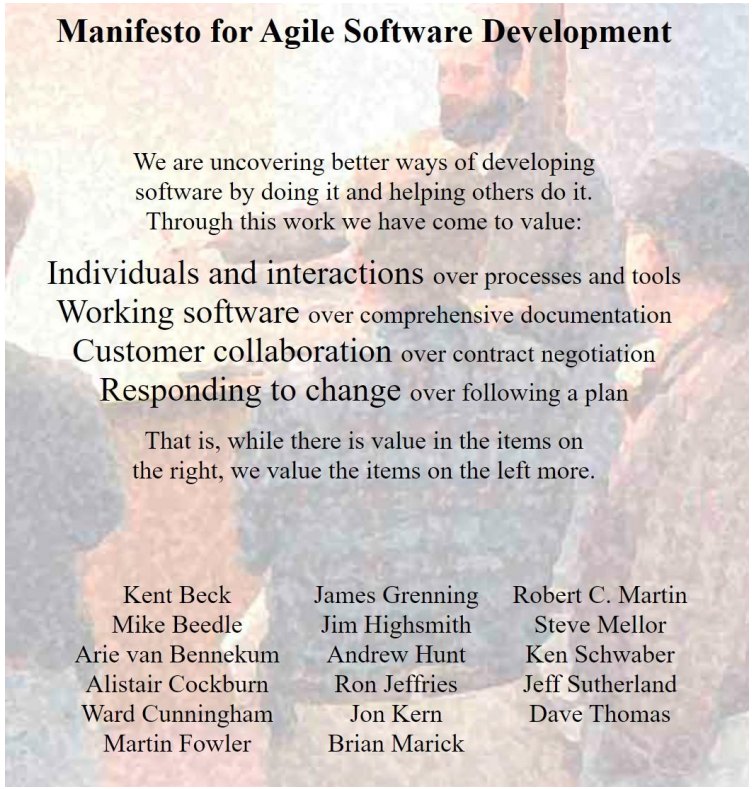


Inputs:

1. Activities - typically by means of WBS (A - H)
2. Duration of each activity (DUR)
3. Dependencies between activities (arrows)

Computed values: Critical path, Earliest / latest start & finish, TF

Agile



- 4 key values and 12 principles (link)
- Created in 2001 as a response to the challenges and limitations of traditional software development methodologies
- It reflected a distillation of ideas and practices that had been evolving within the software development community
- 17 representatives from various agile methodologies came together to articulate a set of guiding principles
  - Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, ..
- The manifesto mentions software development, but the ideas can be applied also to other industries

Agile principles (1)

Develop iteratively and quickly adapt to changes to deliver valuable software

- 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.
- **Working software** is the primary measure of progress

Facilitate communication and collaboration

- **Business people and developers must work together** daily throughout the project.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.

Simplicity (close to the principles of lean thinking)

- **Simplicity**--the art of maximizing the amount of work not done--is essential.

Agile principles (2)

Team empowerment and well-being

- Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done
- The best architectures, requirements, and designs emerge from **self-organizing teams**
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely

Continuous improvement

- At regular intervals, the team reflects on how to become **more effective**, then tunes and adjusts its behavior accordingly

Prioritize technical excellence

- Continuous attention to **technical excellence and good design** enhances agility

Pure agile:

- An approach to agile development that strictly adheres to the core principles and values outlined in the Agile Manifesto

SCRUM

= an agile framework for managing and organizing work, particularly in the field of software development, but also in other industries

- Introduced in the early 1990s

Key principles:

- **Iterative and Incremental**: Scrum breaks down the development process into smaller, manageable iterations (sprints) with a focus on delivering a potentially shippable product increment at the end of each sprint.
- **Collaboration:** Scrum promotes a collaborative and cross-functional approach, with close interaction between team members, the product owner, and stakeholders.
- **Empirical Process Control:** Scrum is based on the principles of transparency, inspection, and adaptation. The framework allows for continuous improvement based on feedback and learning from each iteration.

SCRUM team

Scrum emphasizes the importance of a self-organizing, cross-functional team that collectively possesses the skills and expertise needed to deliver a product increment.

Scrum Master

- Facilitates and supports the Scrum process, removes impediments, and ensures that the Scrum team is working efficiently

Product Owner

- Represents the (business) stakeholders and is responsible for defining and prioritizing the product backlog (the list of features and tasks)

Development Team (3-7 people, including UX, QA)

- Responsible for delivering the product incrementally

What about other traditional roles? - in some cases external roles can be involved, for example

- An analyst when additional expertise on requirements analysis is needed
- An enterprise architect responsible for defining and guiding the overall enterprise architecture in larger organizations or when dealing with complex and interconnected systems
- ….

SCRUM workflow

Sprint: A short time-boxed iteration, typically 2 weeks (no longer than 4 weeks)

Official Scrum ceremonies

- **Sprint Planning:** A meeting at the beginning of the sprint where the team plans the work to be done during the sprint
- **Daily Scrum:** A short daily meeting where the team report progress, set plans for the day, and identifies and addresses impediments
- **Sprint Review**: A meeting at the end of the sprint to review and demonstrate the completed increment to stakeholders
- **Sprint Retrospective**: A meeting at the end of the sprint after the sprint review where the team reflects on the sprint and identifies opportunities for improvement

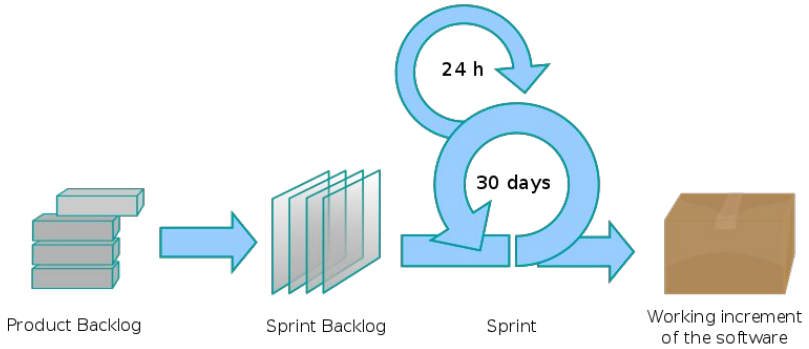
Other meetings

- **Backlog refinement (grooming):** Collaborative sessions to review and prepare product backlog items for upcoming sprints, occur regularly throughout the sprint.
- Release planning. Sprint review preparation, Design / architecture reviews, Technical debt sessions, Dependency review meetings, Daily sync of Scrum of Scrums, …

SCRUM artifacts

Product Backlog

- A prioritized list of all features, enhancements, and bug fixes that need to be addressed in the product
- Backlog items are often in the form of User stories



Sprint Backlog

- The subset of items from the product backlog that the team commits to completing during a specific sprint

Increment

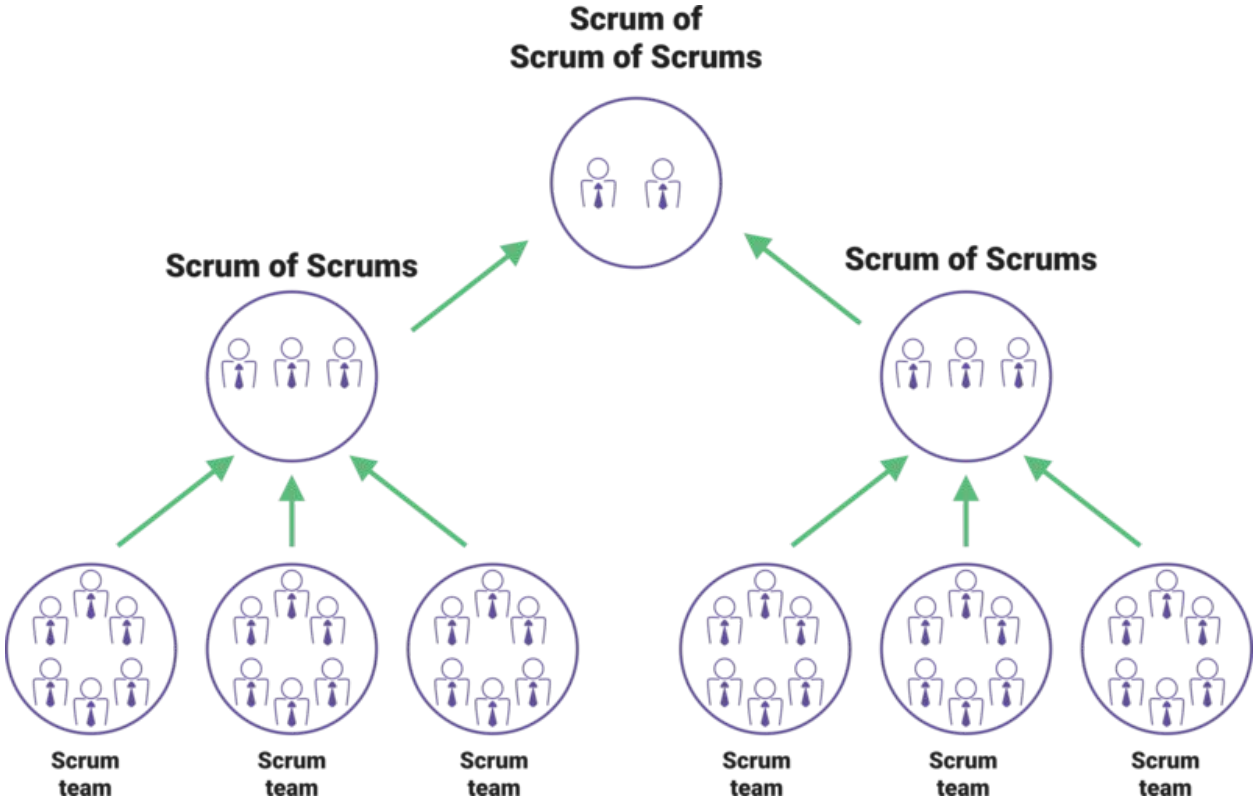
- The sum of all completed items from the sprint backlog, representing the new version of the product

Source: Wikimedia

SCRUM limitations

- Scrum does not cover certain phases of SDLC, for example
  - **Requirement analysis:** Scrum assumes that the product backlog contains the requirements for the product. However, it does not provide detailed guidance on specific requirements analysis techniques.
  - **Architecture planning:** Scrum does not prescribe detailed architectural planning processes.
  - **Deployment and operations:** Scrum does not provide detailed guidance on deployment and operations.
- Scrum cannot be directly scaled to large projects
  - For projects that span multiple products/systems, additional methods need to be used to synchronize different teams, such as
    - Scrum of scrums
    - Large-scale scrum (LeSS)
    - SAFe

Example - Scrum of Scrums



Source: [Wikimedia]

Extreme programming (XP)

= a complete methodology for software development based on agile principles

- Introduced by Kent Beck in the late 1990s (Chrysler Corporation's C3 project)
- Unlike SCRUM, it encompasses both project management and technical practices, aiming for high-quality software.
    - E.g., Test-Driven Development (TDD), pair programming, refactoring, continuous integration
- It uses a combination of short development iterations and continuous feedback to adapt to changing requirements
- Planning is dynamic, with a focus on delivering small, valuable increments
- It emphasizes direct and continuous customer involvement

Why “Extreme”?

- The best practices from traditional software engineering are taken to "extreme" levels

Kanban

= a visual management method for optimizing work processes and improving efficiency

(Kanban means “signboard” in Japanese)

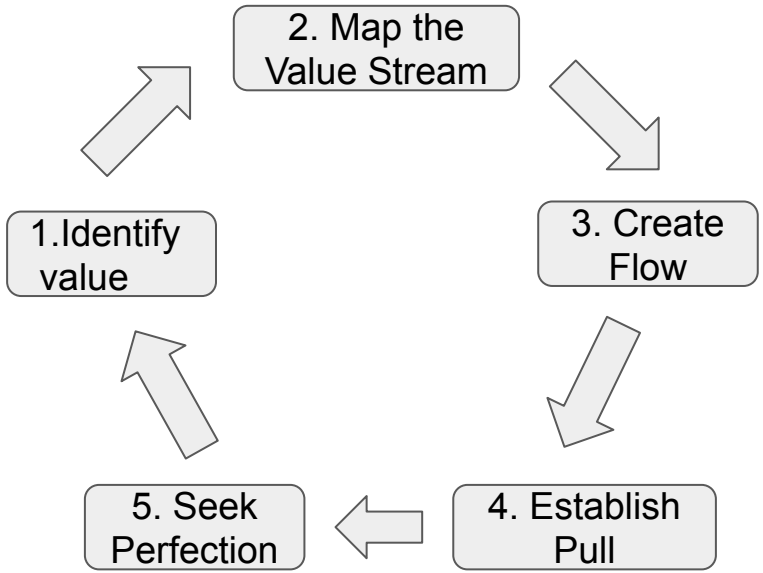
- Originally developed in manufacturing, particularly by Toyota in the 1940s
- Visual boards
  - Represent workflow of tasks or work items

- Work-in-progress (WIP) limits
- Pull system
  - New work is pulled into the system only when there is capacity to handle it
- The goal is to achieve a continuous flow of work through the system
- It often used in conjunction with other agile methodologies and lean principles

Lean management

= a management philosophy and set of principles that aims to maximize customer value while minimizing waste in all aspects of business processes

- Originated from manufacturing practices developed by Toyota in the 1950s
- Applies the five principles of lean thinking to project management
  - **Value:** Identify what customers value
  - **Value stream:** Identify all steps that bring product to the customer. Eliminate steps that do not create value.
  - **Flow:** Turn value stream into a smooth and continuous movement of work.
  - **Pull:** Implement a pull system where work is initiated based on customer demand.
  - **Perfection:** Repeat the process again.



JIRA

- Issue tracking and support for project management
  - Issue = work item to be done / solved
  - Typical attributes: Title, Description, Assignee, Issue type, State, Priority, Relationships to other issues, Start date, Due date, …
  - Workflows (define transitions between issue states)
- It directly supports Scrum, Kanban, but is flexible enough to cover other methodologies or combinations thereof
- Integration with communication platforms (Slack, MS Teams), version control (GitHub) and other SW

- Example: Apache Cassandra Jira

Resources & further reading

- Wikipedia - Critical path method
- Agile manifesto
- Wikipedia - SCRUM
- Ken Schwaber and Jeff Sutherland: The Scrum Guide (PDF)
- SAFe
- SAFe Project Handbook
- Wikipedia - Extreme programming
- Wikipedia - Kanban
- Toyota production system
- Wikipedia - Lean management
- Aziz Moujib: Lean Project Management