Professional Software Engineering

Andrea Carrara and Patrick Berggold

Hritik Singh and Mohab Hassaan – Tutors

Chair of Computational Modeling and Simulation

andrea.carrara@tum.de; patrick.berggold@tum.de

Schedule Lecture 11

- **Software development process**
  - **Software Development Life Cycle (SDLC)**
  - **SDLC methodologies (Waterfall, V, Spiral, Agile)**
- **Software testing process**
  - **Software Testing Life Cycle (STLC)**

andrea.carrara@tum.de; patrick.berggold@tum.de

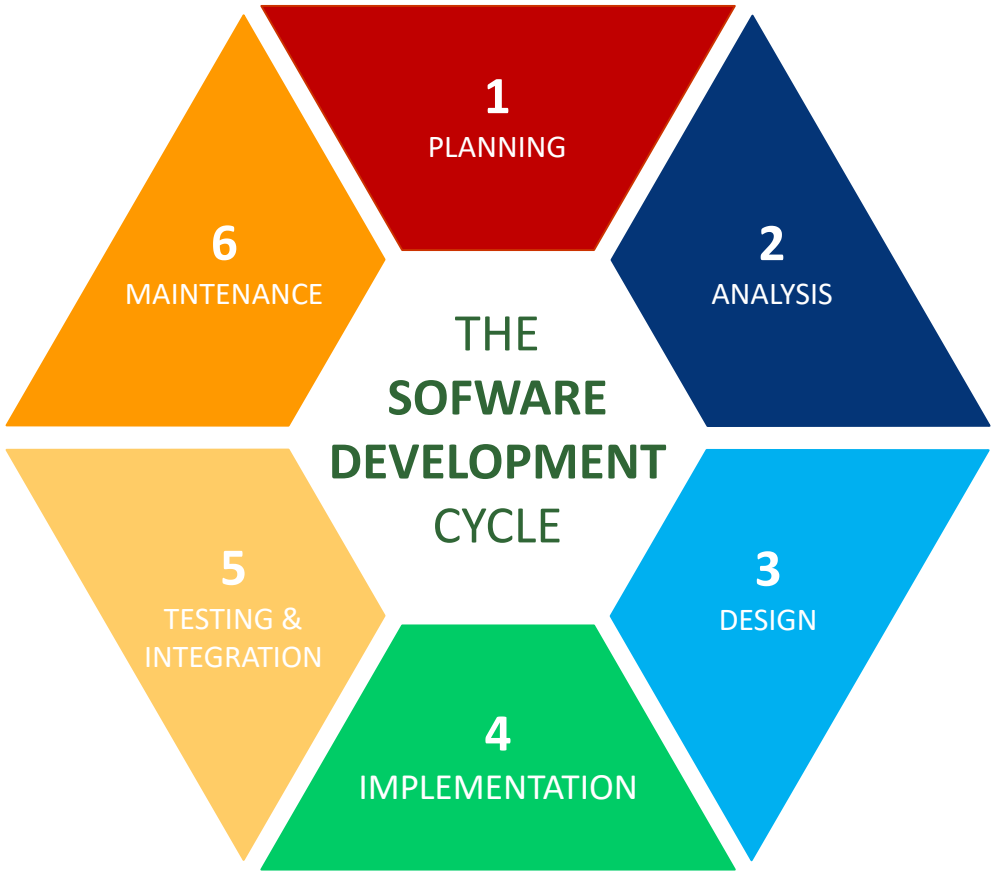
DEVELOPMENT PROCESS

SDLC, Waterfall, V-, Spiral & Agile Methods

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

- Software Development Life Cycle (SDLC) is a set of processes to design, develop and test high quality software.
- It defines tasks performed at each step in the software development process.*
- The SDLC aims to produce high-quality software within times and cost estimates.
- Several SDLC methodologies that implement SDLC differently
  - **Waterfall, V, Spiral, Agile, …**



*number of steps varies from model to model

andrea.carrara@tum.de; patrick.berggold@tum.de

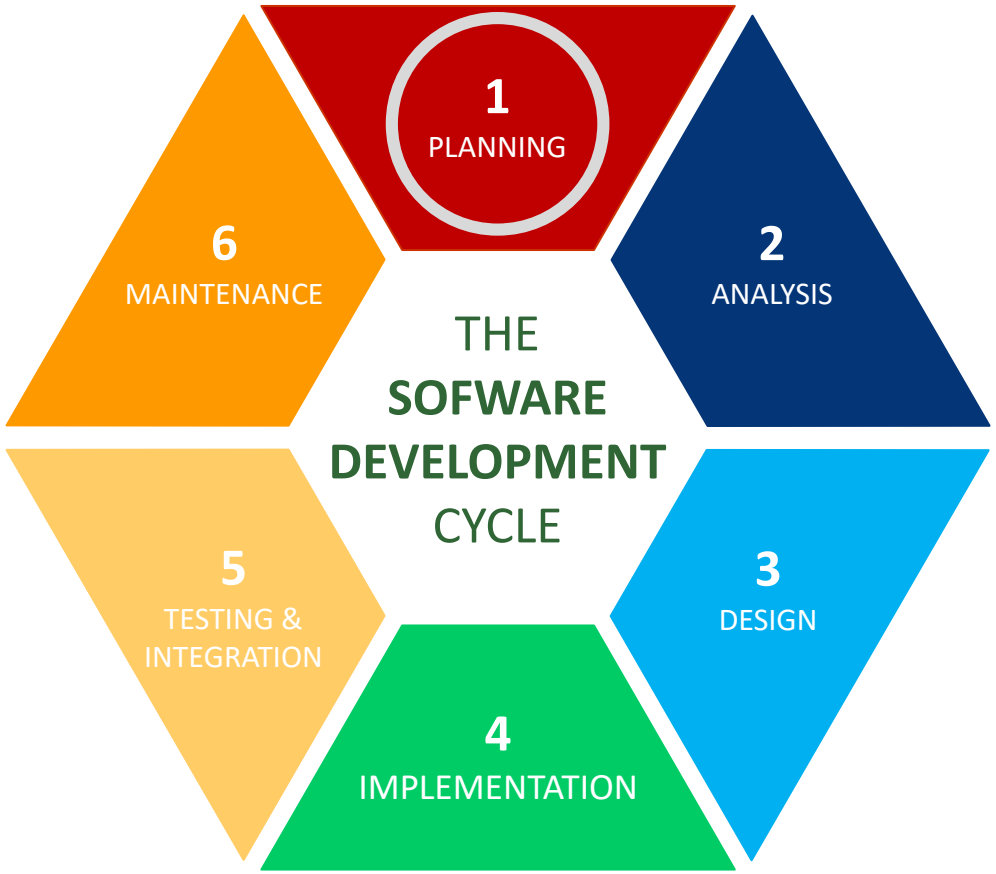
Software Development Life Cycle (SDLC)

Stage 1 – Planning and Requirements Analysis

» Determine the project's requirements
  – **What is the goal?**
  – **Current strengths and weaknesses**
  – **Write from scratch, or upgrade existing system?**
  – **etc.**

» Get input from all stakeholders (e.g. industry experts, programmers and customers)

» Plan the basic project to-do list and approach



andrea.carrara@tum.de; patrick.berggold@tum.de

Requirements Engineering

» **Identify:**

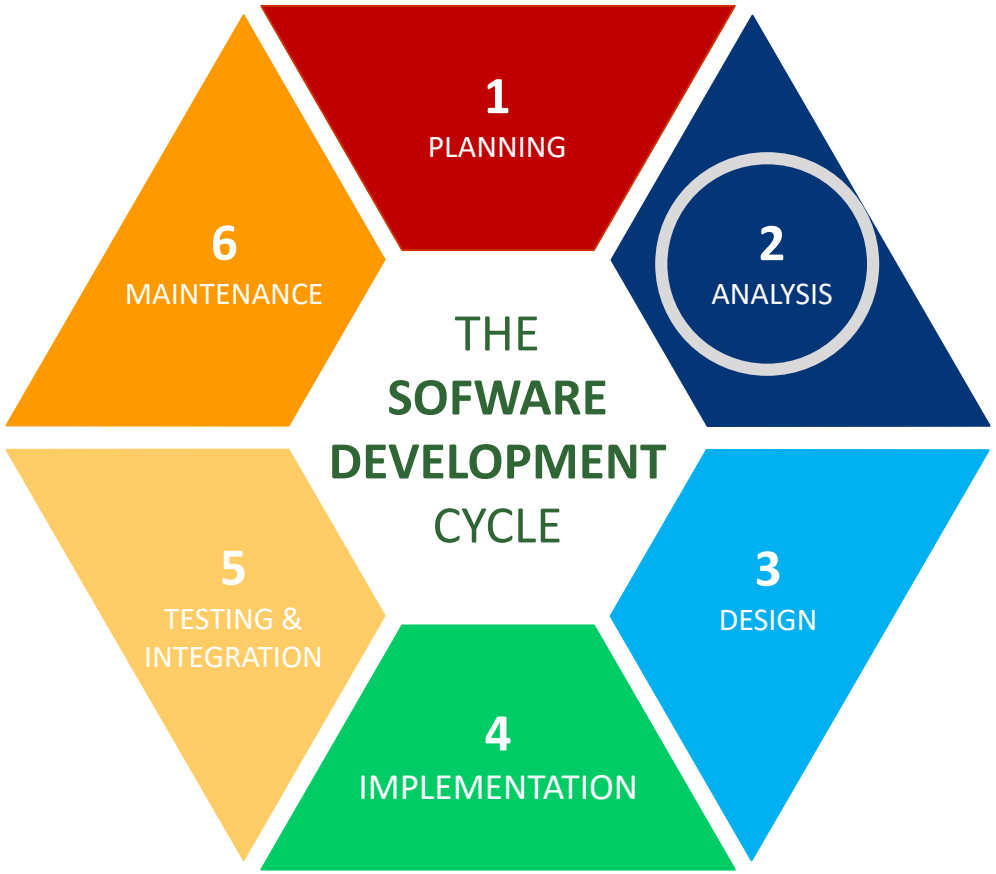
- **Roles** — Who uses this system?
- **Scenarios** — What is the domain? What does the user expect?
- **Use-cases** — Starting & End Condition Process Flow?
- **Refine Use-cases** — Results in a complete system description (system definition, error description, …)
- **Dependency between Use-cases** — Group similar requirements
- **Non-functional requirements** — e.g. Documentation, resources, speed, security, …

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

Stage 2 – Analysis and Feasibility Study

- Document the product requirements, get them approved from the customer or the market analysts
- Determine the project's feasibility:
  - **What costs/resources are required?**
  - **What risks are involved?**
- Requirements and feasibility study must be documented and approved via a SRS (Software Requirement Specification) document



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

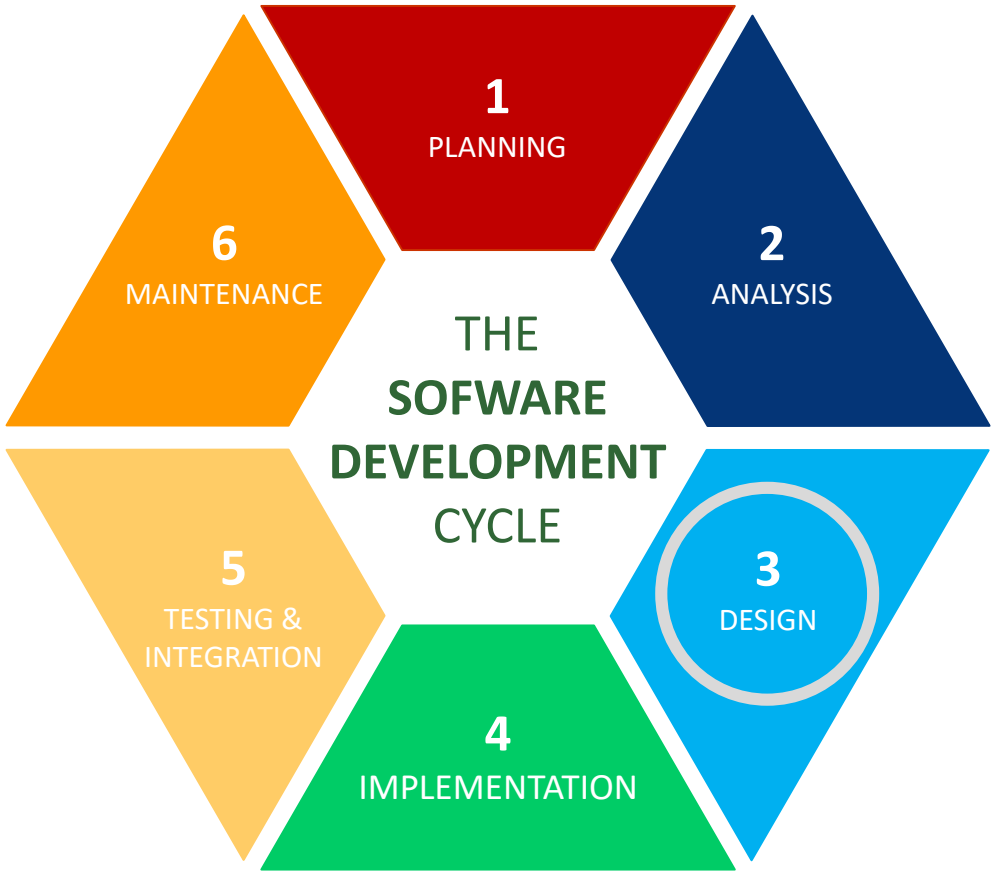
Stage 3 – Architecture Design

» Based on the SRS, the best suitable architecture is planned

» Typically, more than one design approaches are proposed, and documented in the DDS (Design Document Specification)

» The design should define (e.g.) all:

- **architectural modules**
- **communication (input/output, data flow)**
- **external/third-party modules**



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

Stage 4 – Software Development & Build

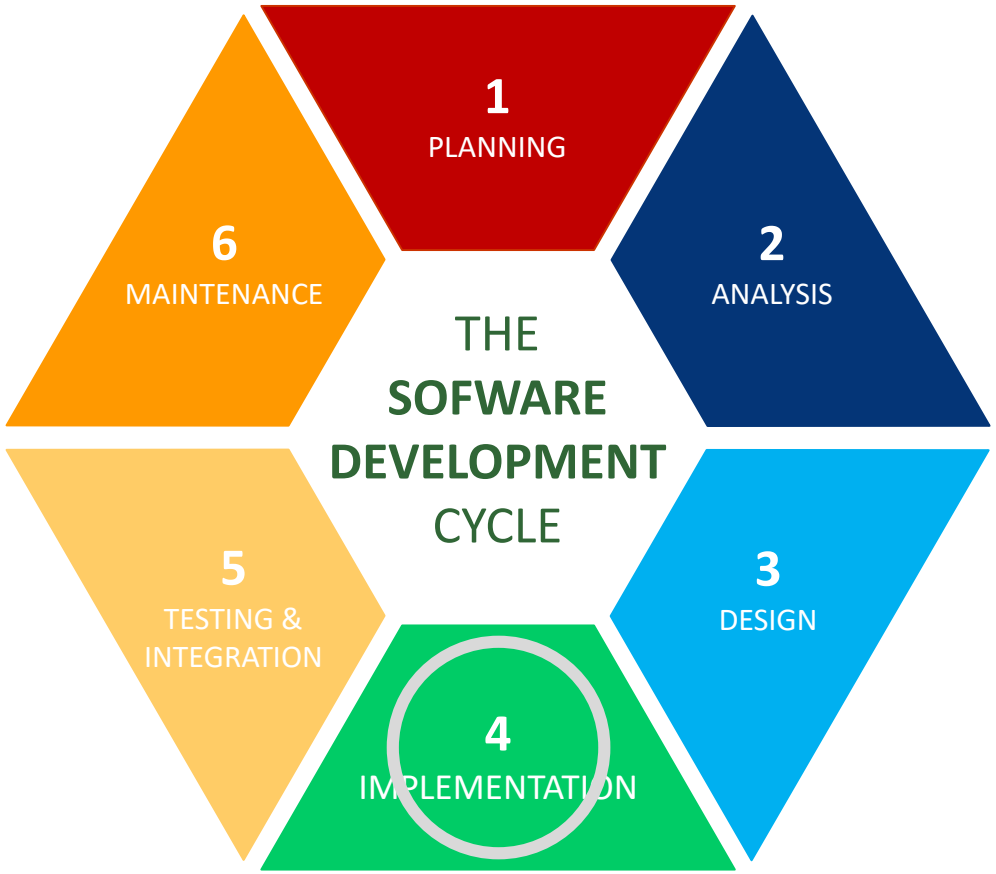
» Based on previous stages, the development team starts working on the code

» Results: Software that meets all the requirements listed in the SRS and DDS

» Proper guidelines should be in-place, e.g.:

- **Code reviews**
- **Nomenclature (e.g. camelCase, under_score)**
- **Which tools to use (compilers, debuggers, interpreters, etc.)**

» Programming language depends on the product



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

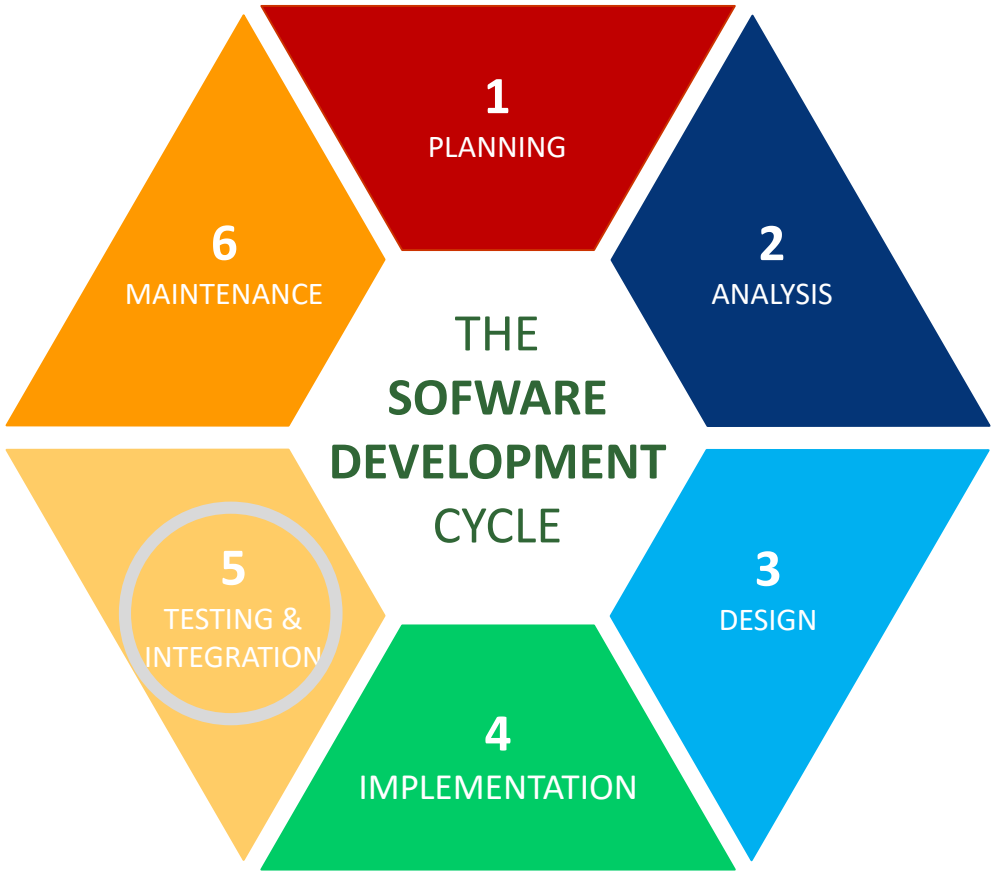
Stage 5 – Testing & Integration

» Extensive testing for defects and deficiencies, and fix issues until the product meets the original specifications.

» Code quality tests involve

- **Unit tests (see Nunit lecture)**
- **Performance tests**
- **Security tests**
- **etc.**

» Outcome: Thoroughly tested version ready for deployment

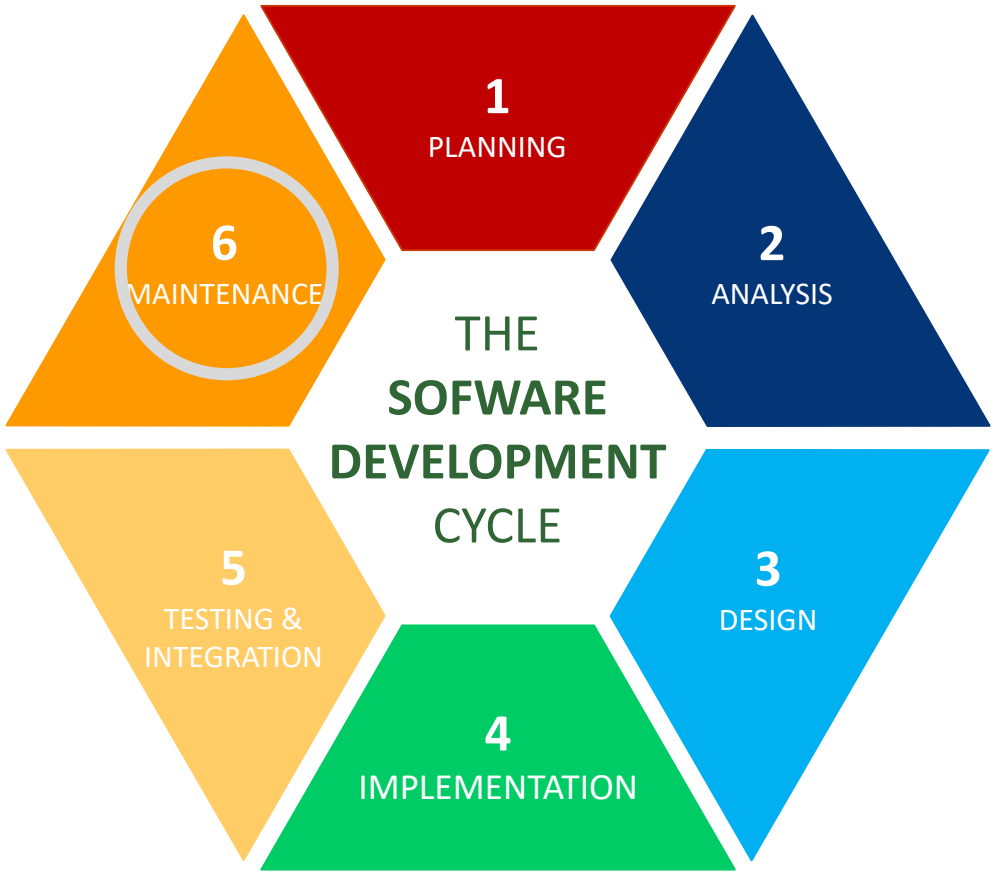


andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

Stage 6 – Deployment and Maintenance

- Software is ready to be deployed either in product environment or testing environment, sometimes with an initially small user base that expands over time
- Maintenance through periodic reviews and updates based on user feedback:
  - **Bug fixes**
  - **Add new features**
  - **Upgrades**
- Whenever bugs or flaws are discovered, product moves back as many stages as necessary



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development Life Cycle (SDLC)

Advantages of SDLC:

- Provides a framework defining activities and deliverables
- Aids with project planning, estimating, and scheduling
- Decreases project risks and overall cost of production
- Increases development speed and visibility on all life cycle aspects to all parties

SLDC methodologies …

- implement the SDLC in different manners
- depend on the project and its objectives
- include amongst others Waterfall, V, Spiral and Agile

Common Pitfalls:

- Misunderstanding of system requirements
- Complexity of the SDLC may cause a project to derail, losing lose sight of specifics and requirements

andrea.carrara@tum.de; patrick.berggold@tum.de

SDLC METHODOLOGIES

Waterfall, V, Spiral & Agile Methods

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Waterfall Model

The **Waterfall** model:

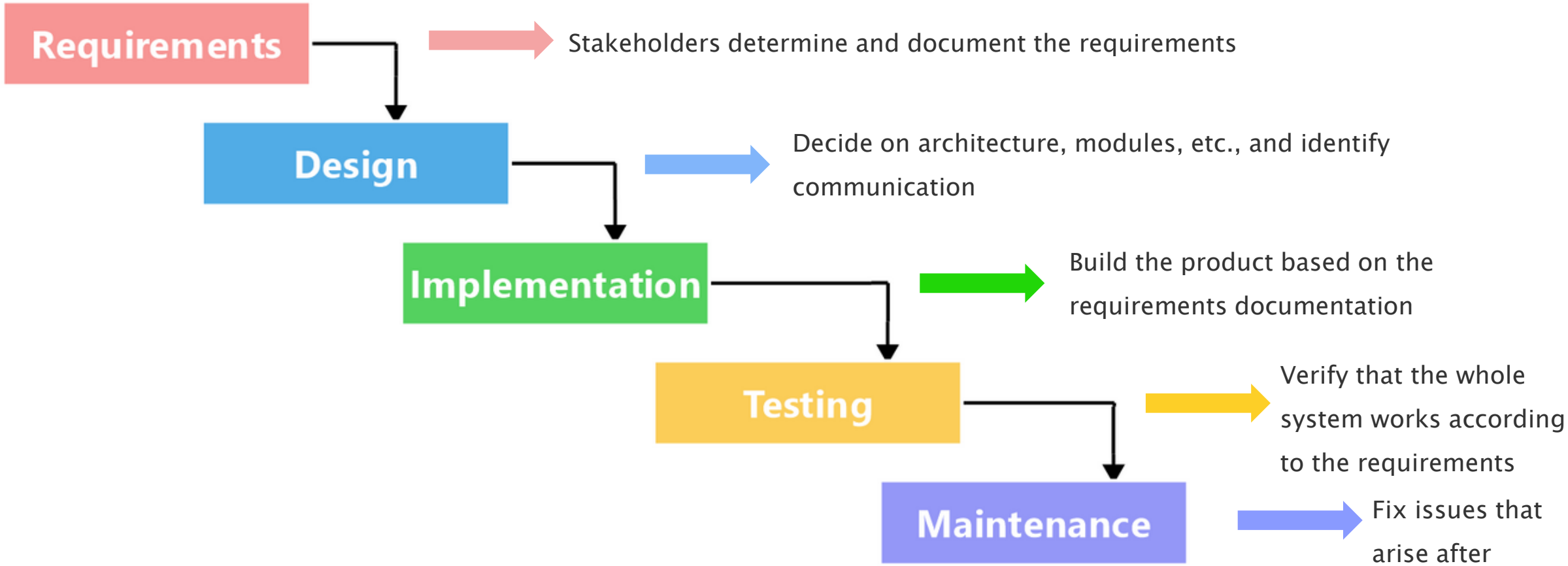
- Sequential, linear: start one phase after succession of the previous
- Each phase depends on the deliverables of the previous (each phase “waterfalls” into the next)
- No overlapping of phases

Use cases:

- Short projects
- Requirements are clearly documented and stable
- Development in non-dynamic, well-understood environment

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Waterfall Model



https://blog.mindmanager.com/waterfall-methodology/

Software Development: V-Model

The V-model:

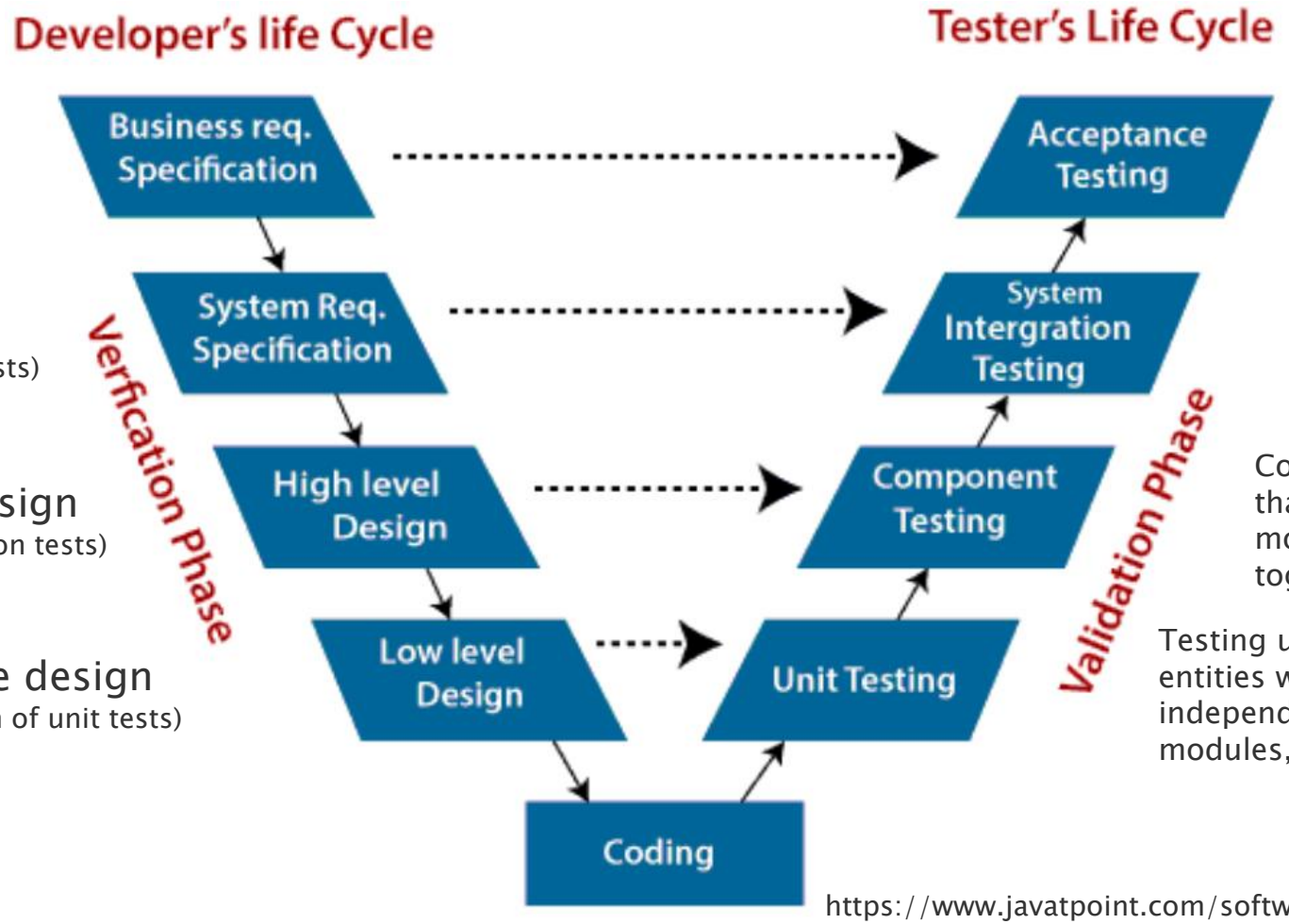
- Also known as *Verification and Validation model*
- Development stages (*verification*) are passed sequentially, during which the corresponding tests (*validation*) are planned in parallel, and if possible conducted
- Extends the Waterfall model by an additional testing phase for each development stage
- After implementation, testing starts sequentially

Use cases:

- See Waterfall model

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: V-Model



System design
(plus design of system tests)

Architecture design
(plus design of integration tests)

Module design
(plus design of unit tests)

Verify entire system functionality

Component test verifies that independently tested modules can function together

Testing units, the smallest entities which can independently exist (e.g. modules, functions)

https://www.javatpoint.com/software-engineering-v-model

Software Development: Waterfall & V-Model

Advantages:

- Easy control, scheduling and overview
- Workload clearly separated and defined
- Process and tasks are well documented

Disadvantages:

- Working software is produced only late during the life cycle
- Project success highly dependent on initial requirements
- Too rigid for complex, long-lasting projects

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Spiral Model

The **Spiral** model:

- A software project repeatedly passes through four repetitive phases (called spirals)
- Continuous prototyping coupled with systematic progress
- Thus, software is released incrementally

Use cases:

- For high-risk, complex or large projects
- In dynamic environments (e.g. changing technologies, markets)
- Unclear or complex requirements, goals, use cases

andrea.carrara@tum.de; patrick.berggold@tum.de

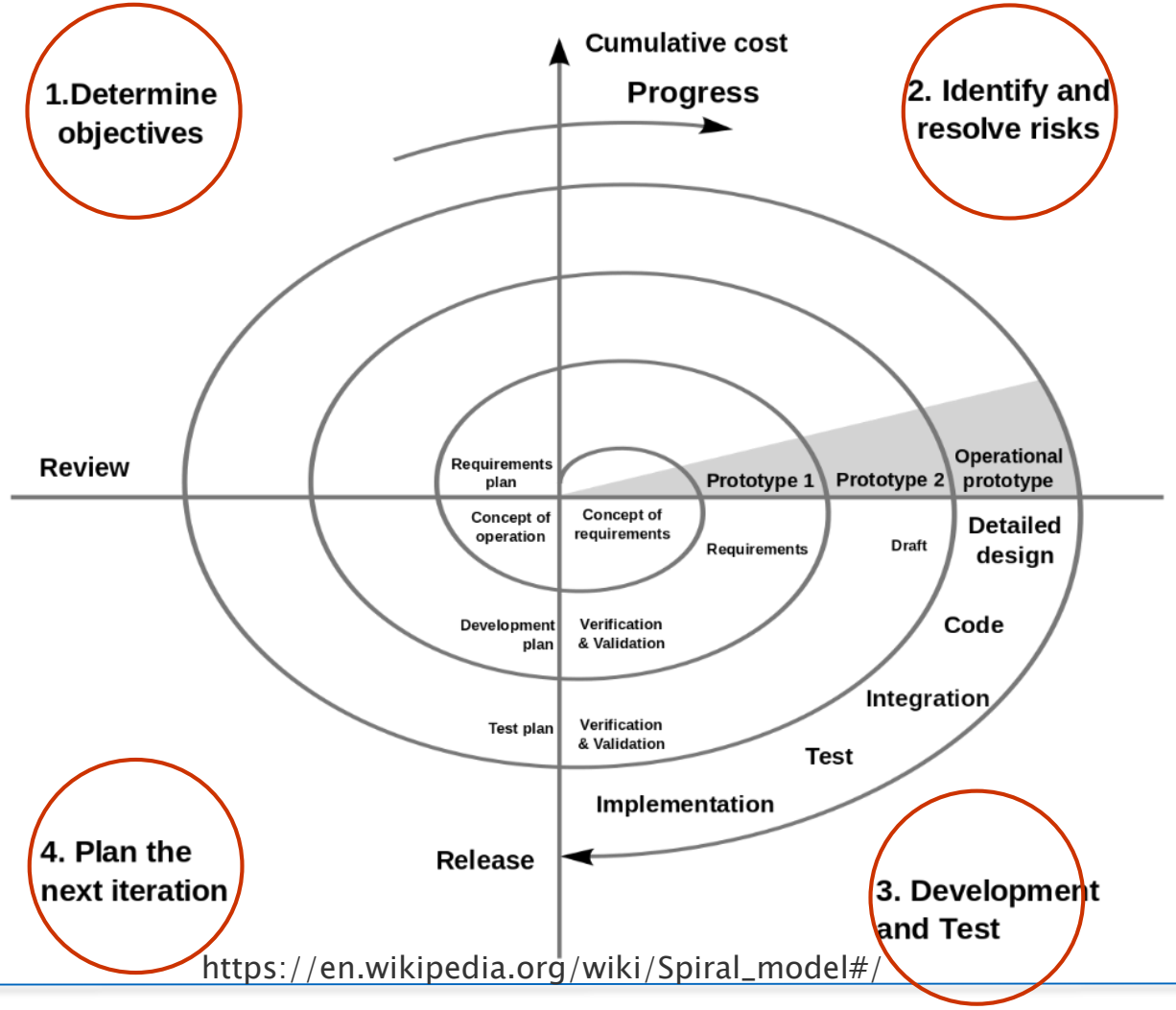
Software Development: Spiral Model

Planning:

- Establish the goal of that cycle, explore alternatives
- Identifying the according requirements (business, system, subsystem, unit)

Evaluation:

- Assess results of previous spirals
- Evaluate software (what is still missing) based on the project's review by customer



https://en.wikipedia.org/wiki/Spiral_model#/

Risk Analysis:

- Estimate technical feasibility and risks (regarding technical, scheduling, cost aspects)

Engineering:

- Development, integration and testing of the actual software (proof-of-concept, prototypes, final release)

Software Development: Spiral Model

Advantages:

- Early phases of the project already produce prototypes
- Product flaws can be identified and mitigated early
- Goals, requirements and risks are re-assessed repeatedly

Disadvantages:

- Management and documentation can be confusing and complex
- For small projects unnecessarily time- and cost-consuming

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile Model

The **Agile** model:

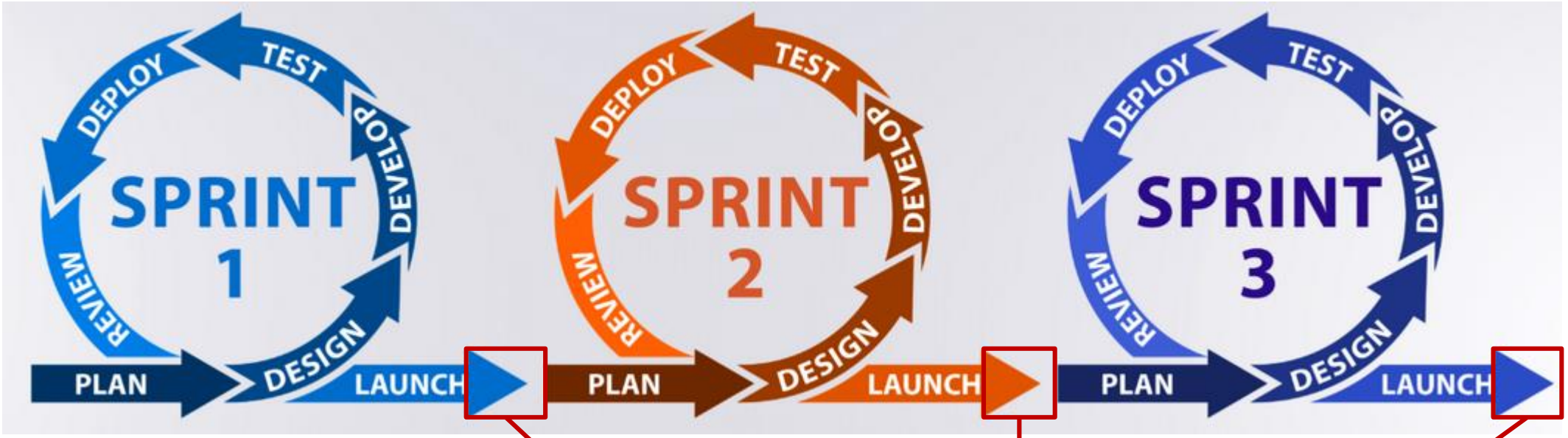
- Breaks tasks into smaller iterations (called sprints) of flexible duration (approx. 1 week – 1 month)
- Project's requirements and sprints (number, durations, tasks) planned at beginning of the project
- Sprints are individually tailored and adapted for a specific task (extensions or shortenings are possible)
- Sprints carried out by small teams (of specialists) that intensely collaborate

Use cases:

- Small or medium size projects (some frameworks also handle large-scale well, e.g. Scaled Agile Framework)
- When requirements may change, high uncertainties
- Tight schedule, customer needs immediate results

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile Model



https://kruschecompany.com/de/agile-softwareentwicklung/

Software Development: Agile Model

Advantages:

- High flexibility, quick results ➔ suitable for dynamic environments (markets, technologies, etc.)
- Close contact to clients typically leads to high satisfaction rate
- Early software generation leads to early identification and mitigation of design issues
- Minimal overhead, little documentation

Disadvantages:

- Experienced and highly specialized team required
- Little documentation may lead to neglect of maintenance
- Intense sprints wear a team down over time

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile vs. Spiral

Agile vs. Spiral:

- Sprints vs. spirals
  - **Sprints are highly flexible, can happen very differently with different goals depending on each sprint's goal**
  - **Spirals undergoes the same repetitive pattern**
- Documentation: little vs. high
- Risk handling/analysis: little vs. a lot
- Typical team size: small vs. medium
- Suitable project size: small/medium vs. complex/large
- Customer interaction: high vs. occasionally

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: The Agile Manifesto

The Agile Manifesto

1. Customer satisfaction by **early and continuous delivery** of valuable software.
2. Welcome changing requirements, even in late development.
3. **Deliver working software frequently** (weeks rather than months)
4. **Close, daily cooperation** between business people and developers
5. Projects are built around **motivated individuals**, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. **Working software is the primary measure of progress**
8. **Sustainable development**, able to maintain a constant pace
9. **Continuous attention** to technical excellence and good design
10. **Simplicity**—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from **self-organizing teams**
12. Regularly, the **team reflects on how to become more effective**, and adjusts accordingly

» Created in February 2001 in Utah by 17 software developers

» Encourages collaboration and the sharing of ideas, gets rid of documentation

https://en.wikipedia.org/wiki/Agile_software_development

Software Development: Agile Frameworks

Agile software development has further lead to various popular practices, or Agile methodologies

- **Examples: Scrum, Kanban, eXtreme Programming (XP), Crystal, … (more than 50)**

Difference between Agile and Scrum/Kanban/XP/…:

» Agile is a philosophy or a set of principles, while frameworks implement the Agile philosophy

» These frameworks are specific methods for managing the development process based on the Agile manifesto

- **… providing roles, events, and artifacts that teams can use to plan, execute, and review their work**

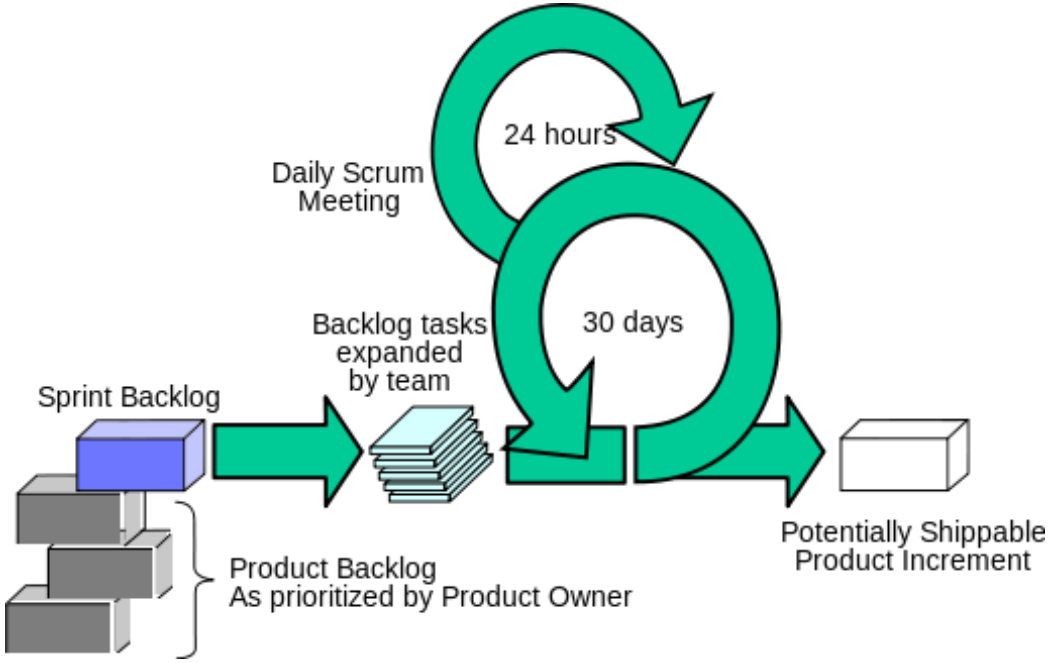
» Agile is a way of *thinking,* Scrum/Kanban/XP/… are a way of *working*

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile Frameworks: Scrum

Scrum is by far the most commonly used Agile framework

- Sprints are ~14 days
- Three different roles:
  - **Scrum Master**: coach, mentor, organize and support the team
  - **Product Owner**: defines and prioritizes the product backlog
  - **DevTeam**: manages and organizes work to complete the sprint
- Includes a product backlog
  - **Prioritized list of features and requirements that the product needs to include**

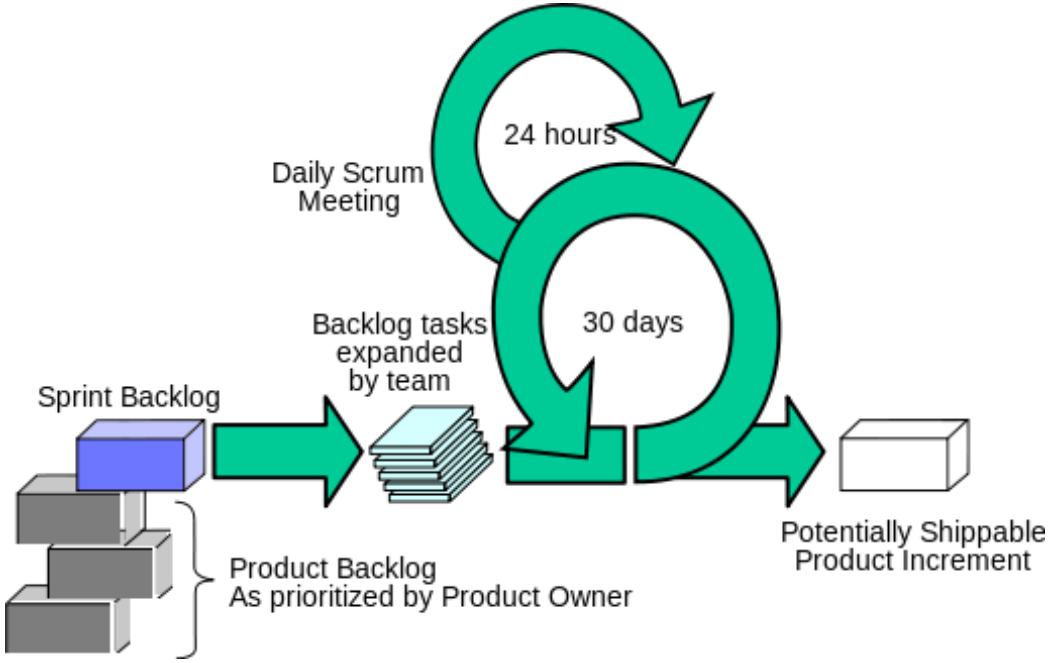


andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile Frameworks: Scrum

» Sprint Backlog plan for next sprint

– **List of items from the product backlog that the DevTeam commits to completing during the current sprint.**



5 Scrum ceremonies:

1. Sprint planning (before sprints)

   Product Owner and DevTeam decide on which items from product backlog to include in the sprint backlog, and what will be done during the sprint, and how

2. Daily scrum

   Discussing progress and next steps, and the encountered obstacles daily

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile Frameworks: Scrum

3. Backlog refinement (frequently)

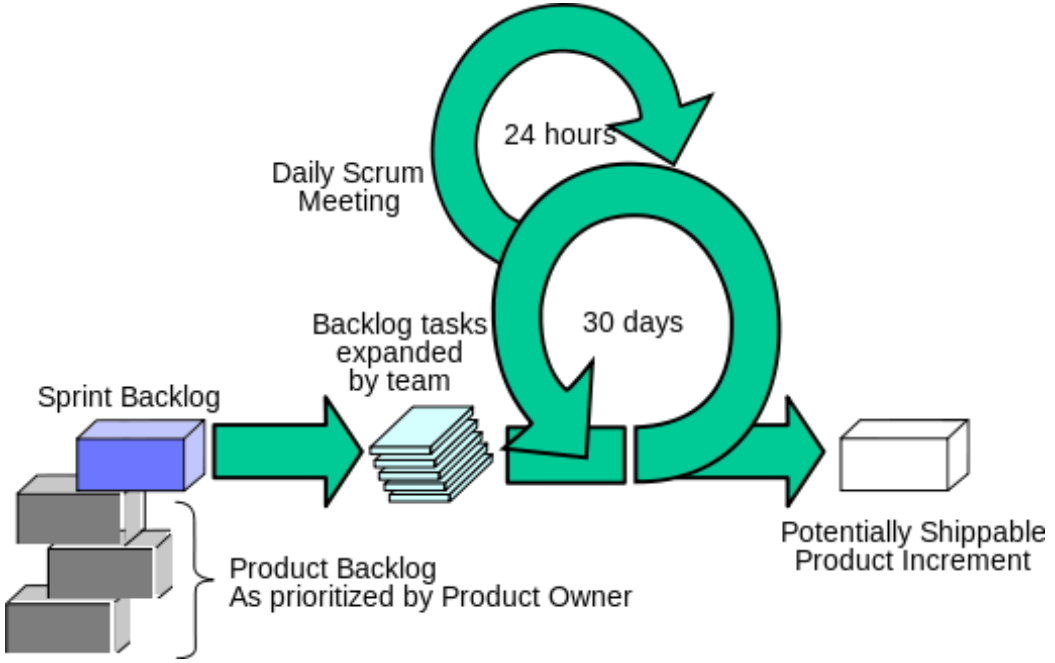
   Ongoing process where the team reviews, prioritizes and adds items in the product backlog

4. Sprint retrospective (after sprints)

   What can be improved (communication, tools, processes, etc.), no focus on the product directly

5. Sprint review (after sprints)

   After each sprint, get feedback from stakeholders and ensure that requirements are met, demonstrate functionality



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Agile Frameworks – Popularity

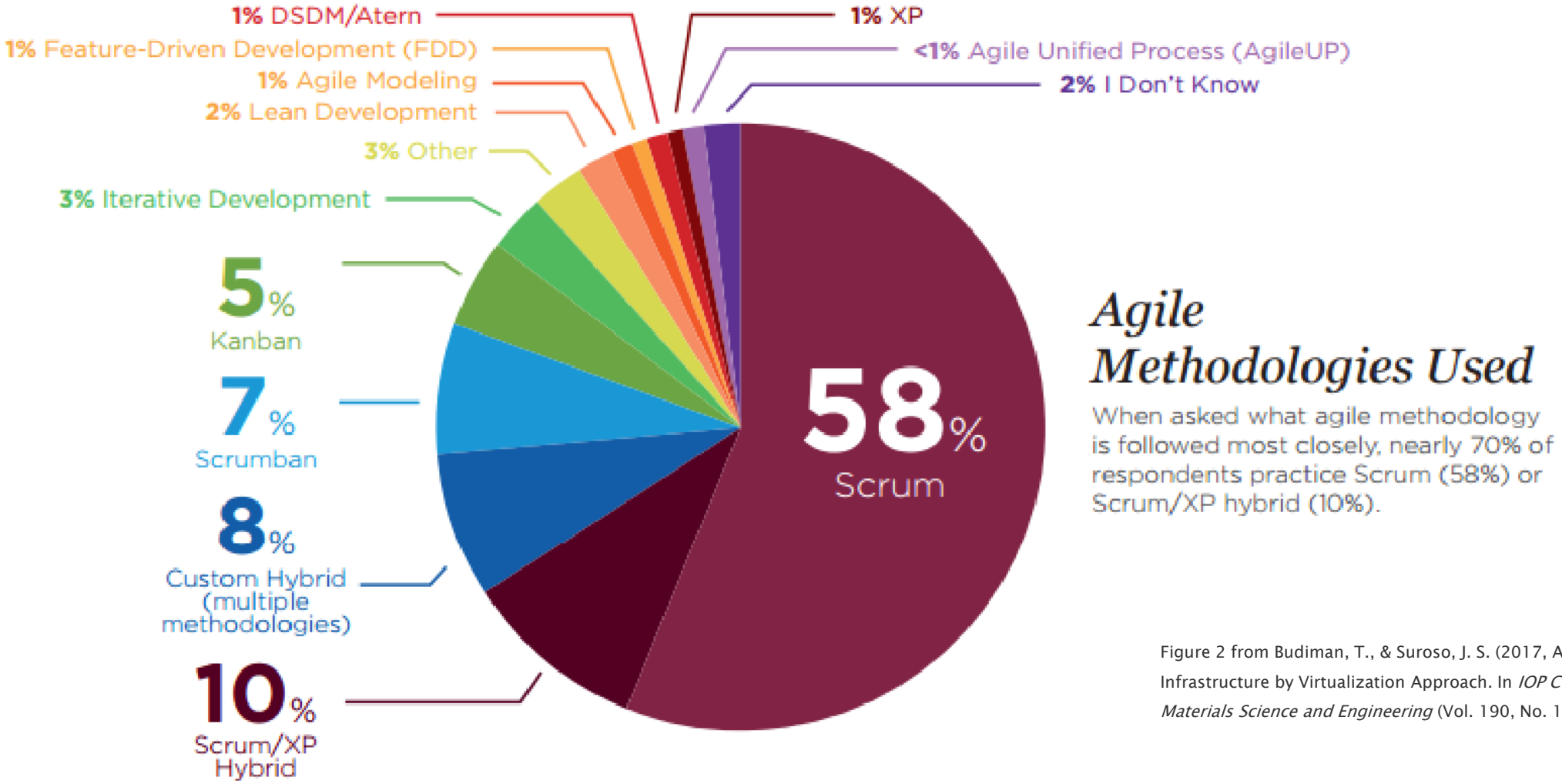
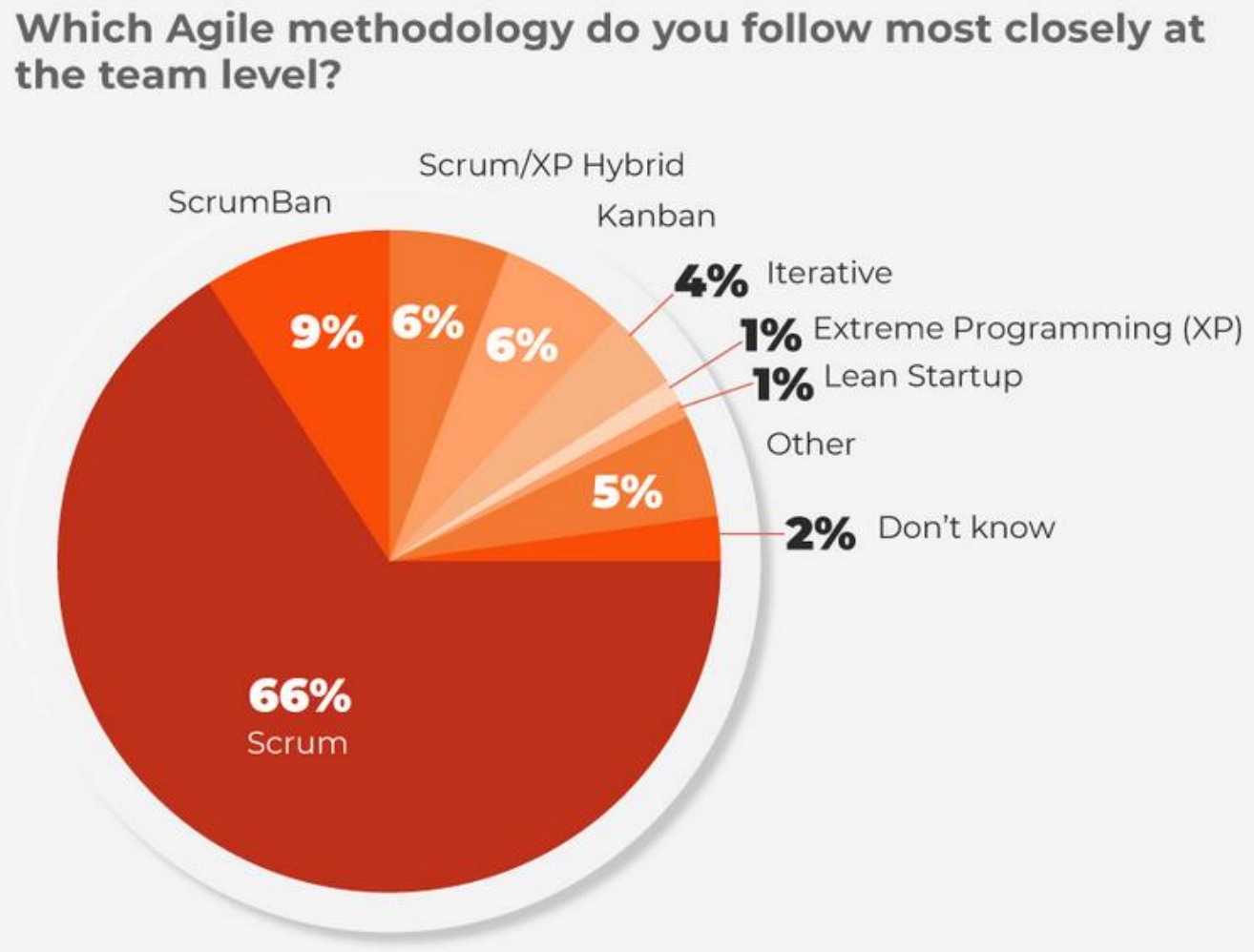


Figure 2 from Budiman, T., & Suroso, J. S. (2017, April). Optimizing IT Infrastructure by Virtualization Approach. In *IOP Conference Series: Materials Science and Engineering* (Vol. 190, No. 1, p. 012012).

Software Development: Agile Frameworks – Popularity



from the 15th State of Agile report, 2021, https://itnove.com/wp-content/uploads/2021/07/15th-state-of-agile-report.pdf

Software Development: Agile Frameworks – Scrum Popularity

Why is Scrum so popular?

- Variety of certifications
  - **Scrum Masters, Product Owners, developers, trainers, coaches can get certified by the Scrum Alliance**
- Simple to understand and implement
- Frequent collaboration with customer
- High visibility into the project progress through the Scrum ceremonies

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Development: Comparison Waterfall vs. Spiral vs. Agile

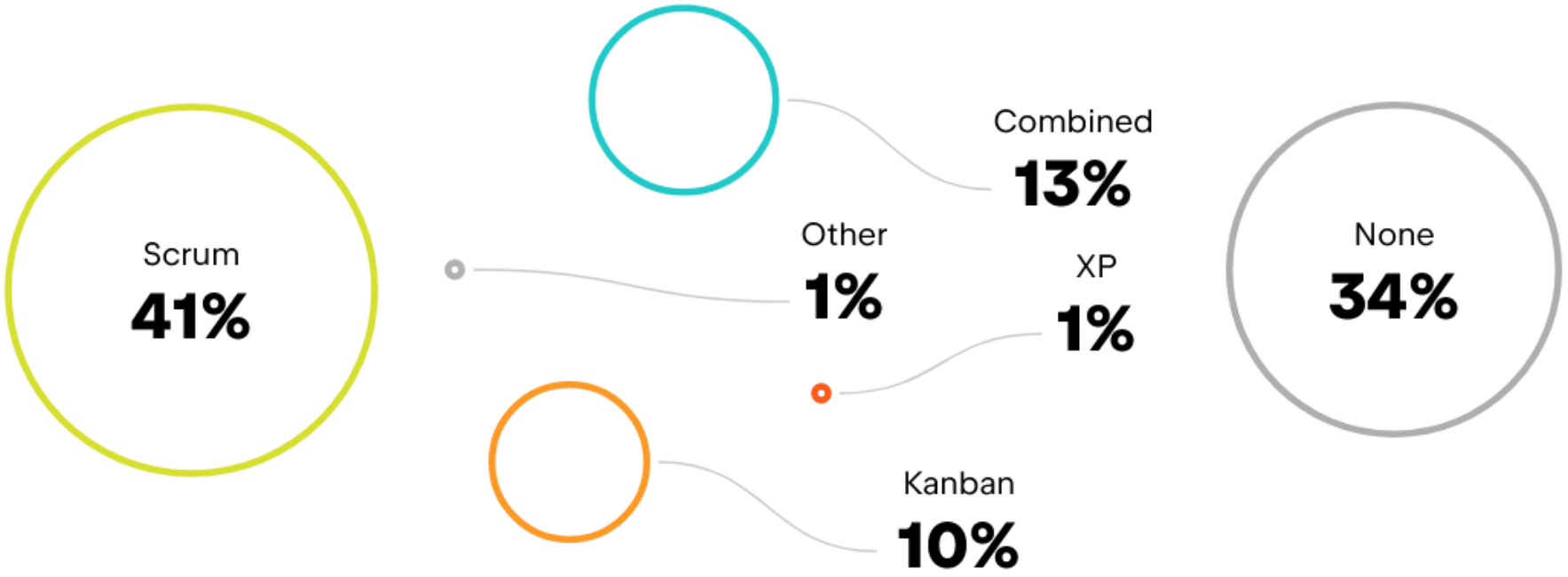
S. No	Points	Waterfall	Spiral	Agile
1	Method	Sequential method.	Evolutionary method.	incremental & Iterative
2	Customer	Easy to understand	Tough to understand	Easy
3	used for Type of project	Small	Large	Large
4	Risk identification	Later stage	Earlier stage	Every stage
5	Flexibility	Difficult to adopt changes	Easy to change requirement	Changes accepted at any stage
6	RISK OF USE	Higher	Lower	LOWER
7	SIMPLICITY	Simple	Complex	EASY
8	CUSTOMER INVOLVEMENT	Less	less	MORE
9	DEADLINES	Large	large	SHORT
10	CLARITY IN REQUIREMENT	Beginning	Beginning	REQUIREMENT DYNAMIC
11	COST	Fixed	May change with process	FLEXIBLE
12	TASK	Phases	Iterations	SPRINTS
13	FOCUS	Project delivery	Project delivery	CUSTOMER SATISFACTION
14	TESTING	After development stage	IN EACH ITERATION	IN EACH ITERATION
15	DEPENDENCY	Project manager	Project manager	SCRUM MASTER
16	TEAM SIZE	Large	Medium	SMALL
17	DOCUMENTATION	More	More	LESS

Kodmelwar, M. K., Futane, P. R., Pawar, S. D., Lokhande, S. A., & Dhanure, S. P. (2022). A Comparative Study of Software Development Waterfall, Spiral and Agile Methodology. Journal of Positive School Psychology, 6(3), 7013–7017.

From *A Comparative Study of Software Development Waterfall, Spiral and Agile Methodology*

Software Development: SDLC methodology popularities

What agile software development framework do you use in your team, if any?



Jetbrain's "State of Developer Ecosystem Survey in 2018"

andrea.carrara@tum.de; patrick.berggold@tum.de

TESTING-PROCESS

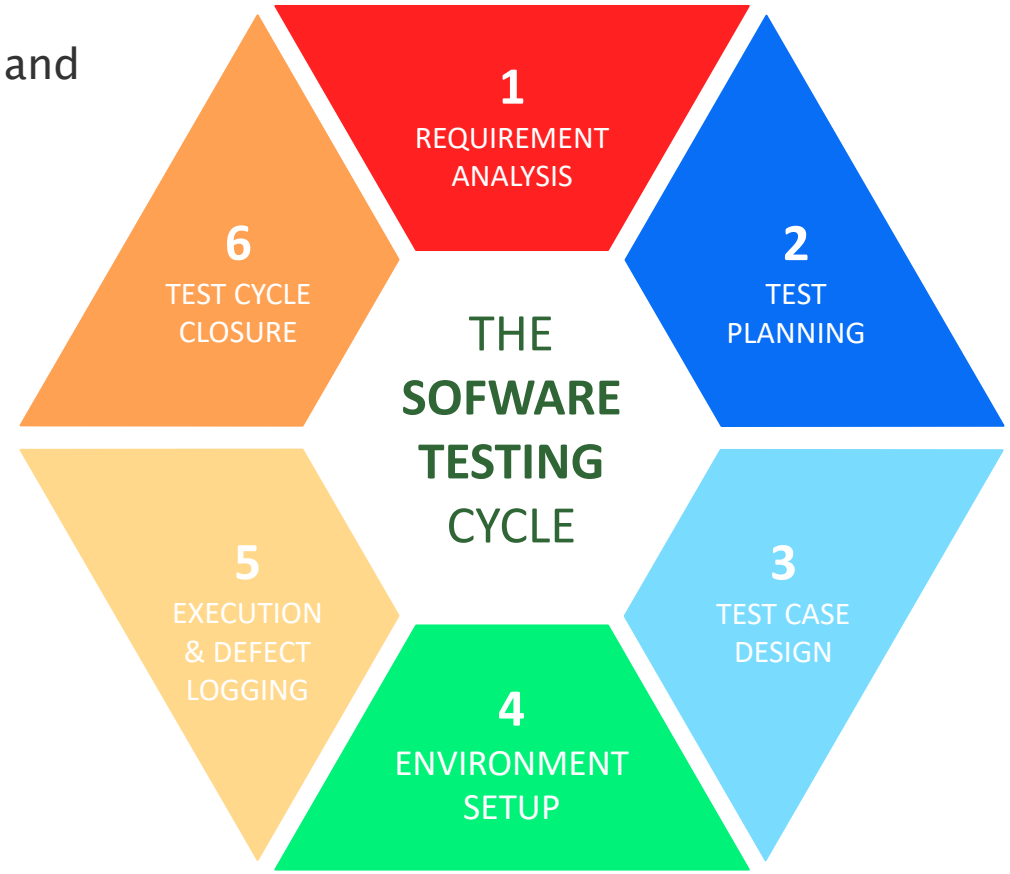
STLC

andrea.carrara@tum.de; patrick.berggold@tum.de

Software Testing Life Cycle

- Software Testing Life Cycle (STLC) is a set of processes to test and evaluate high quality software.
- Solely dedicated to testing software

- Describes the stages involved in the testing of software to...
  - **... provide a framework and processes to thoroughly test software**
  - **... to improve quality of software and ensure that it meets the requirements before it is deployed.**



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Testing Life Cycle

1. Requirements analysis

» Test team analyses the requirements document from SDLC, the SRS

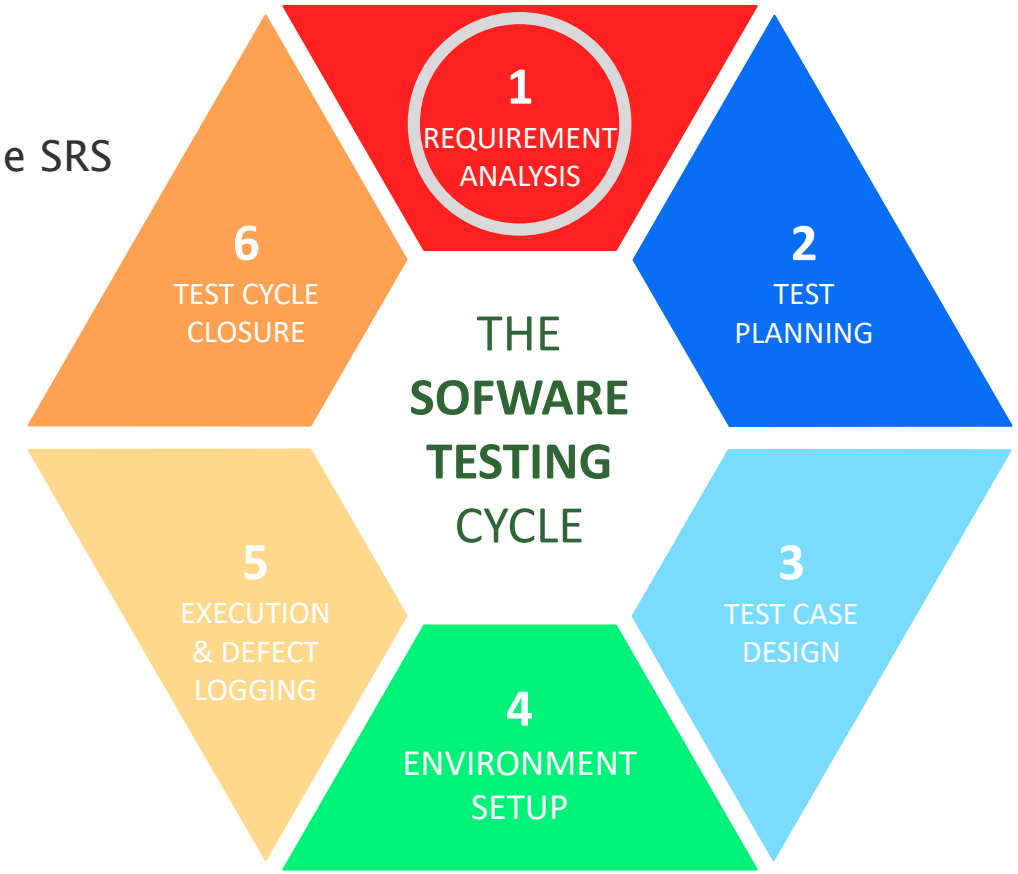
» Identify testable requirements, close in contact various stakeholders

» Some goals of this phase:

– **Identifying blind spots or unclear areas in the requirements**

– **Identify types of tests and test environments**

– **Prioritize certain assessments**



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Testing Life Cycle

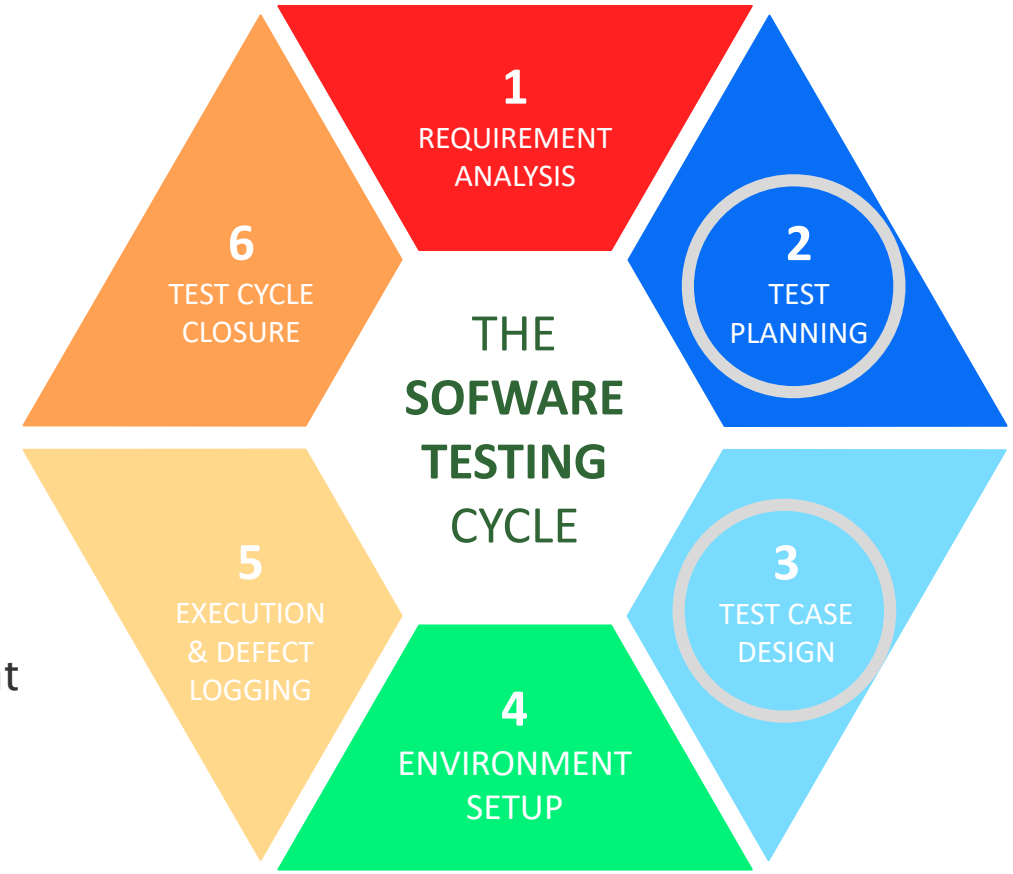
2. Test planning

» Test plan is prepared and finalized:

- **Which costs, efforts, resources?**
- **Determine priorities, tools, responsibilities and scheduling**
- **Which limitations?**

3. Test case design

» Create test cases and scripts corresponding to the requirement analysis



andrea.carrara@tum.de; patrick.berggold@tum.de

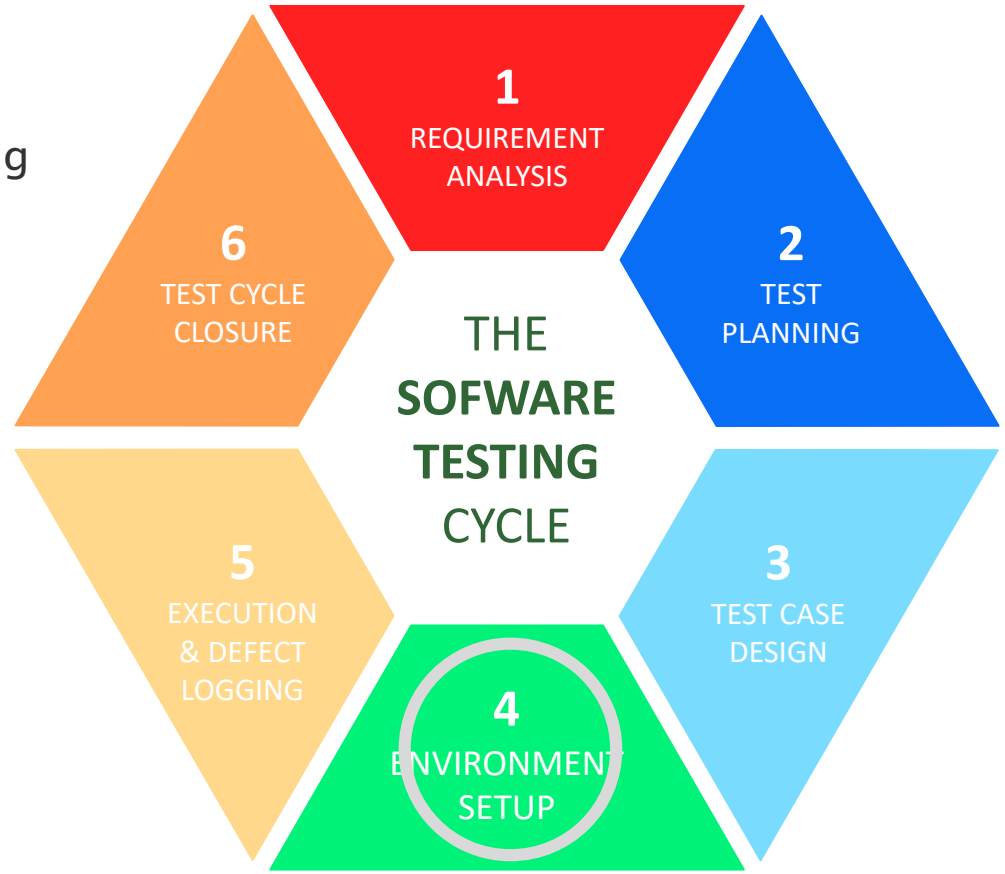
Software Testing Life Cycle

4. Environment setup

» Environment provides the surrounding setting where the testing occurs

» For the environment, important parameter like hardware, software, test data, etc. must be established, e.g.:

- **Device (Laptop, work station, phone, ...)**
- **Operating system (Windows, Linux, Android, iOS, ...)**
- **Browser (Firefox, Chrome, Edge, ...)**
- **Processing (CPU, GPU, ...)**



andrea.carrara@tum.de; patrick.berggold@tum.de

Software Testing Life Cycle

5. Execution & Defect logging

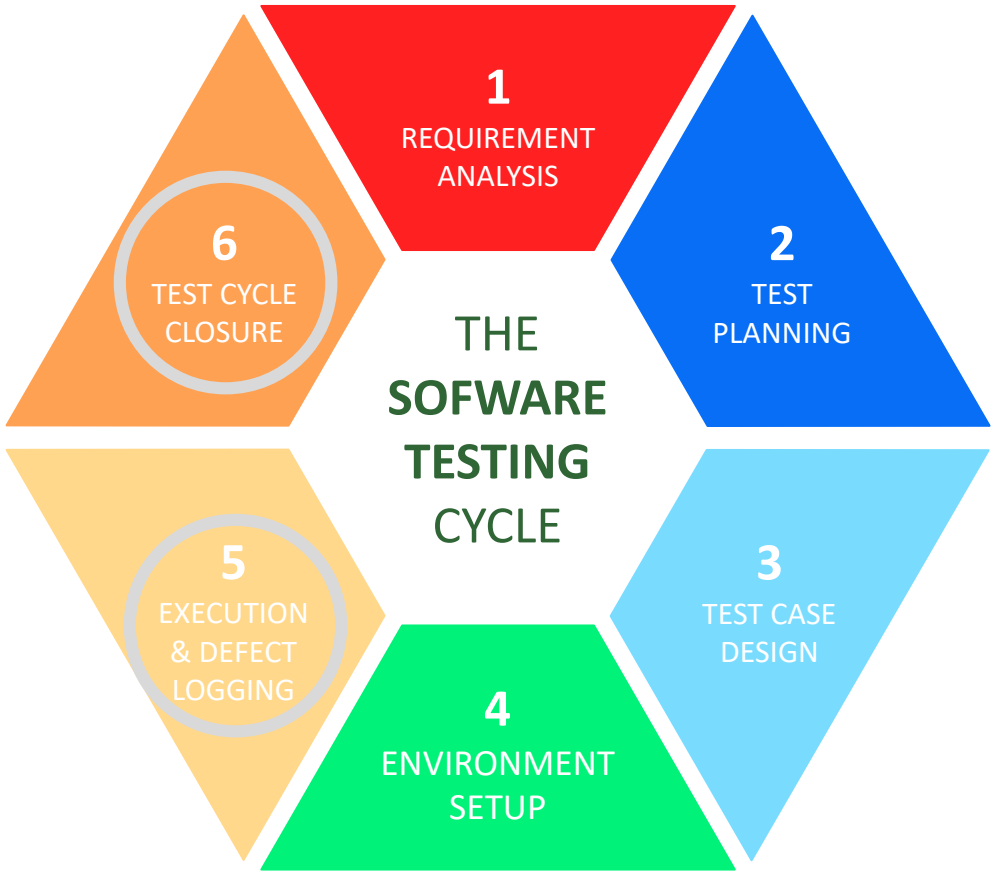
» Execute the planned tests in the setup environment

- **Identify and report bugs, flaws, etc.**
- **Log the system's performance compared to its requirements**
- **Developers make fixes, not the testers!**

6. Test cycle closure

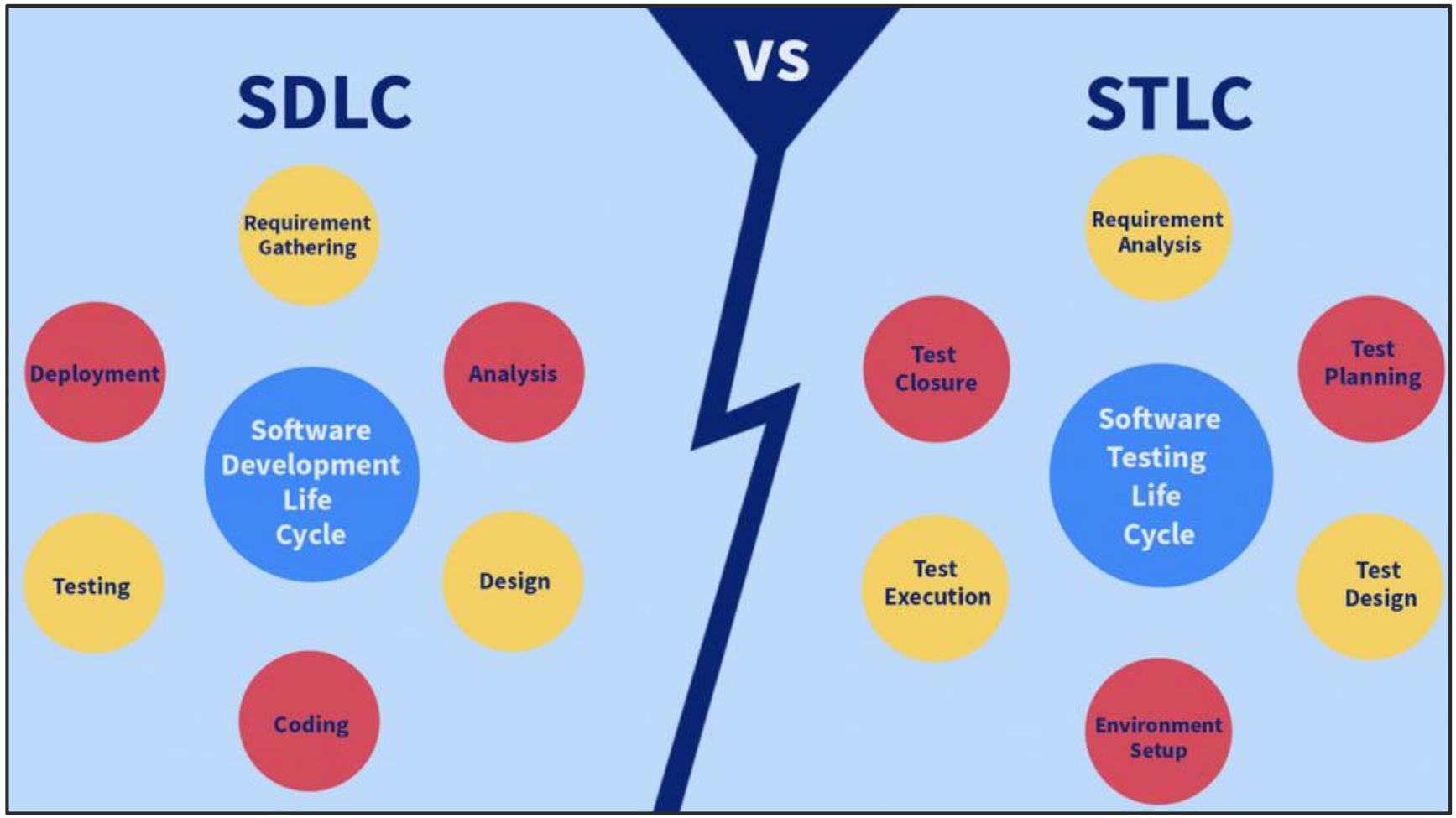
» Finalize and document test report, analyze the test results thoroughly, set up test completion matrices

» Use the gained knowledge for future tests and projects



andrea.carrara@tum.de; patrick.berggold@tum.de

STLC and SDLC



https://www.interviewbit.com/blog/sdlc-vs-stlc/

STLC and SDLC

- Testing is an important **part** of the software development process
- Thus, STLC is a part or subset of SDLC
- SDLC provides the overall framework, whereas STLC complements this framework with detailed testing activities
  - **(partially) overlapping timeline (depends on the methodology)**

- Similar set of processes for maintenance is called Software Maintenance Life Cycle (SMLC)

andrea.carrara@tum.de; patrick.berggold@tum.de

THANK YOU!

andrea.carrara@tum.de; patrick.berggold@tum.de

Sources

https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/

https://en.wikipedia.org/wiki/Systems_development_life_cycle

https://phoenixnap.com/blog/software-development-life-cycle

https://stackify.com/what-is-sdlc/

https://www.synopsys.com/glossary/what-is-sdlc.html

https://www.tutorialspoint.com/sdlc/index.htm

andrea.carrara@tum.de; patrick.berggold@tum.de