Agile Development in Cloud Computing Environments

Information Technology (M.Eng.)

Module 11: Optional Technical Subject

SoSe 2022

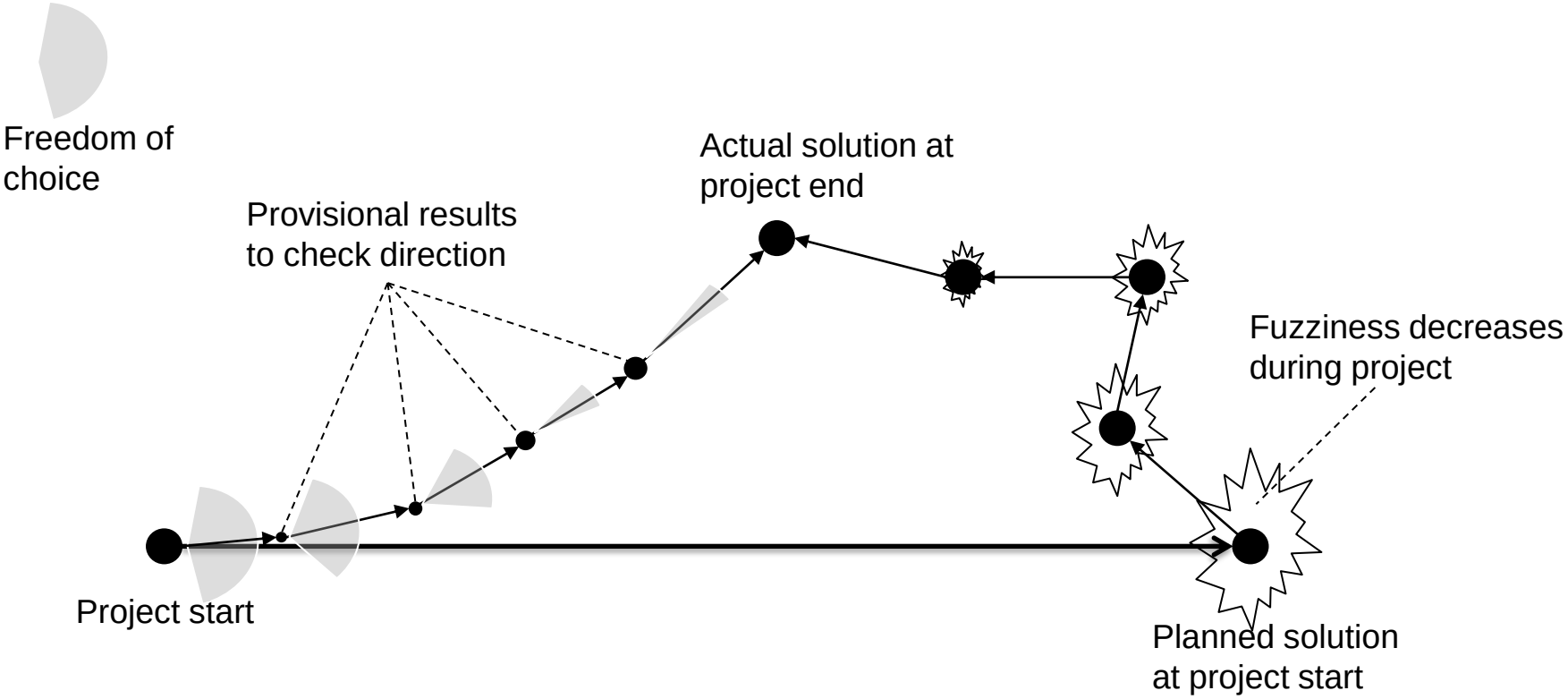
Dr. Patrick Wacht (email: patrick.wacht@fb2.fra-uas.de)

Frankfurt University of Applied Sciences, Frankfurt a. M., Germany

General information

- What will you learn in the course?
  - Agile development methodology – theory and practice/experience
  - Deepen knowledge in software development
  - Learn how to work with a leading cloud computing environment/platform
- What are the key objectives?
  - Learn about how real-world projects can be done using agile methods
  - Improve skills in the area of
    - Team work
    - Project and time management
    - Negotiations, assertiveness and techniques of scientific work

Problem of uncertainty and risk in software projects



[Starke, 2002]

Where does the uncertainty come from?

- **Customer** does not know what we wants (maybe just at the moment)
- Not all requirements and stakeholders are known
- **Customer** has contradictory requirements (e.g. due to political reasons)

- **Contractor** does not exactly understand what the customer wants
- **Contractor** underestimates / overestimates expenditure (planning)

- Changes in the priorities or business processes during the project
- Project is embedded within a complex project landscape

- Technical risks (e.g. infrastructure does not meet the requirements as expected)

How do traditional methodologies deal with the problem?

Question 1:

What kind of traditional methodologies or rather process models to you know for software development?

Question 2:

How do they work?

Question 3:

Have you already gained experience with these process models? What is your opinion regarding the feasability?

How do traditional methodologies deal with the problem?

- **Requirements**
  - Coordination takes place completely at the beginning of the project
  - Definition of a precise system specification (contract basis)
  - Changes only possible via a predefined Change Request process
  - Targets → exact understanding of the requirements, stability of requirements (traceability)

- **Architecture**
  - Development in early project phase (min. coarse architecture)
  - Taking into consideration requirements of later phases
  - Targets → stable framework of the software

- **Documents**
  - Everything is documented, especially the specification and architecture documents
  - For Offshore partners, operations teams, development teams
  - Targets → Clarity, review capability or rather stability and reduction of communication

How do traditional methodologies deal with the problem?

- **Risk management**
  - Projects are regularly performed in a few steps (3 – 12 months)
  - Active (oftenly unsystematic) handling of risks (existing list of risks)
  - Risks are oftenly considered superficially

- **Plans**
  - Coarse planning at the beginning of the project (due to pricing), detailed planning if required
  - If time elapses → unimportant functions or test expenses are discarded
  - Targets after priority: 1. Meet the deadline 2. Keep to the budget 3. Quality 4. Complete functionality

- **Contracts**
  - Ofentimes fixed price projects
  - Major customers usually have a purchasing department which decides by price

How do traditional methodologies deal with the problem?

- **Customer**
  - Integrated depending on availability (at the beginning of the project permanent, also "on-site")
  - Involved via requirements definitions, document reviews and system and acceptance tests
  - Communication oftenly regulated by project manager (SPOC→ Single Point of Contact) or via documents
  - Targets → long-term customer loyalty, trust, binding agreements

- **Processes**
  - Activities and workflows are defined in detail
  - Many roles involved (e.g. in RUP / V-Model up to 30)
  - Many precisely defined artefacts (documentation, source code…)
  - Targets → Predictability, repeatability, optimized (following CMM model → "Capability Maturity Model")

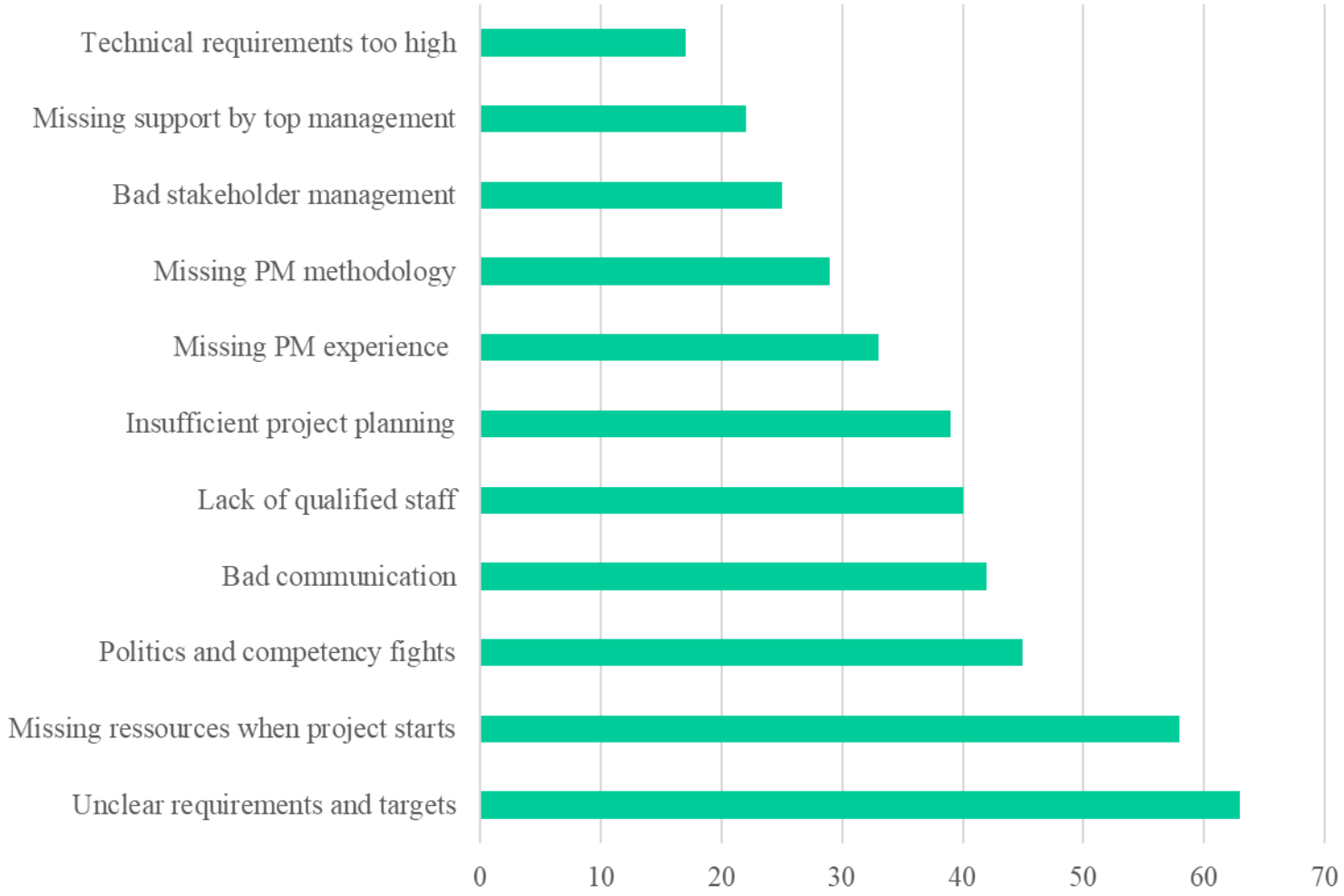
Criticism on traditional methodologies

- Changing of requirements and opinions is forbidden → very expensive!
- "Software bureaucracy": too many documents are created, especially reports concerning speficiations and architecture
- Late feedback of the customer ("Waterfall issue")
- Customer does not get what he wanted – he gets what is specified in the contract
- Stimulates very complex software designs (Chief Architect monuments)
- Neglects competencies of developers
- Too expensive, too sluggish

Traditional methodologies can be used when…

- contractors (developers) can easily estimate the requirements
- the requirements do not change
- the projects are big (large-scaled projects)
- a high security in the development is required ("over"-planning necessary)

Why do projects fail?



[PA, 2019]

Aims of agile methodologies

- Higher **flexibility** than in traditional methodologies
- Focussing on the **targets** that need to be **achieved**
- Solving of technical **and** social problems
- Avoiding of bureaucracy
- Fail fast – fail cheap – **fail early**

Manifesto for agile software development (Feb 2001)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to the value:

1. Individuals and interactions **over** processes and tools
2. Working software **over** comprehensive documentation
3. Customer collaboration **over** contract negotiation
4. Responding to change **over** following a plan

That is, while there is value in the terms on the right, we value the items on the left more.

[Beck et al., 2001]

12 Principles behind the manifesto

1. Satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer‘s competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

[Beck et al., 2001]

12 Principles behind the manifesto

7. Working software is the primary measure of progress.

8. Agile processes promote sustanable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity – the art of maximizing the amount of work not done – is essential.

11. The best architecture, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts ist behaviour accordingly.

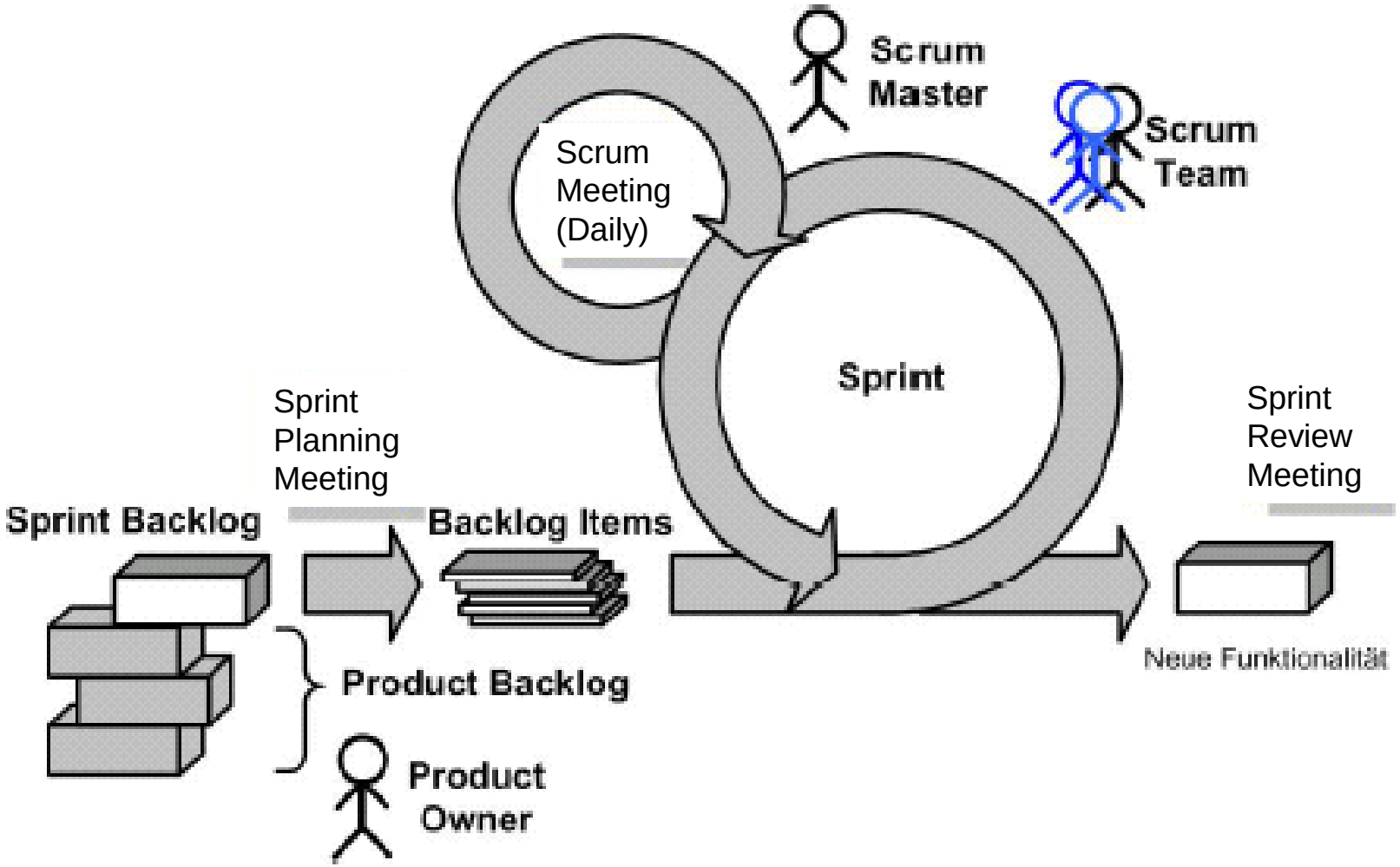
[Beck et al., 2001]

How is the manifest implemented?

Through agile methods, such as:

- Crystal Clear, -Yellow, -Orange, -Red
- Adaptive software development (ASD)
- Scrum
- Extreme Programming (XP)
- Test-driven development (TDD)
- …

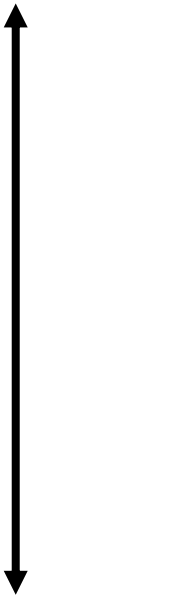
Example: Scrum



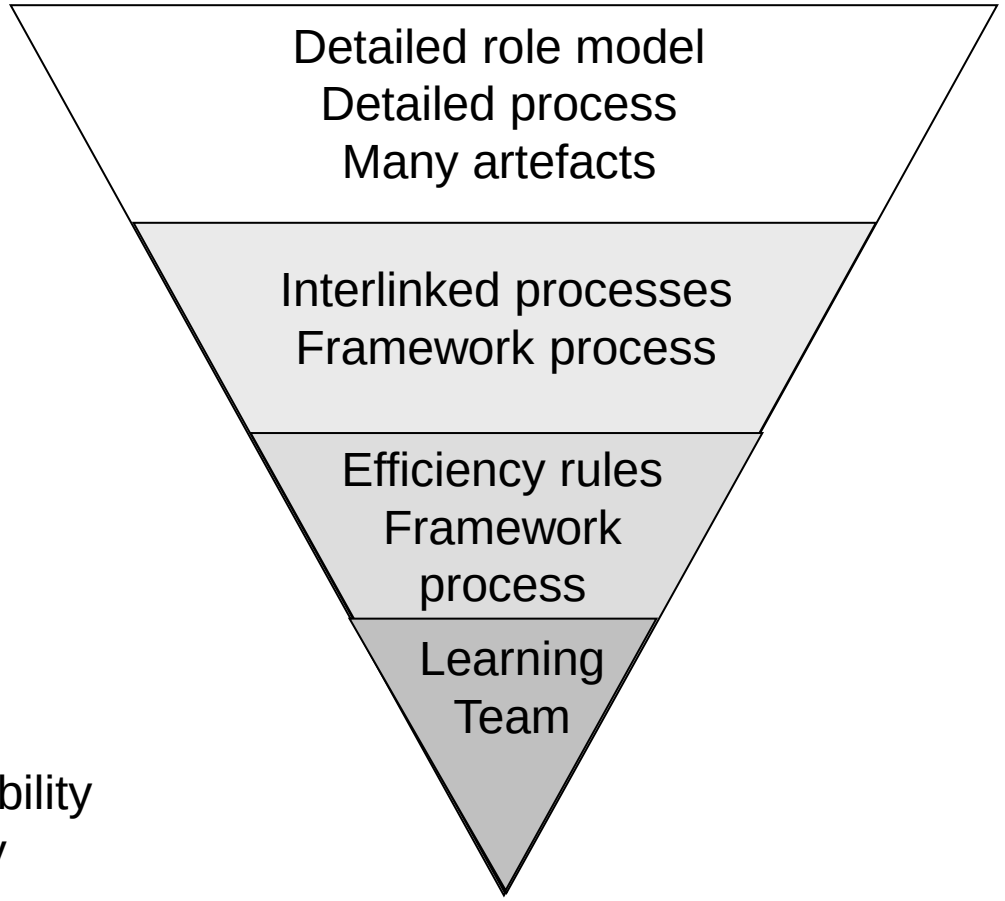
[Schweizer, 2003]

Process vs. Responsibilities in agiles methods

More detail



Individual responsibility
Team responsibility
Flexibility



(CMM conform)
V-Modell (XT)
RUP

Crystal-Orange

Extreme
Programming

Crystal-Clear

Scrum

ASD

[Beck et al., 2001]

Summary

- Agile methods
  - Different perspective on developing software than traditional methodologies (such as Waterfall, V Model or RUP)
  - Can save expenses if applied correctly
  - Faster time to market

- Agile methods = different culture, diverent value system
  - Acceptance of management, purchasing department, customer, developer

- Agile methods are not always the best choice
  - Big teams, critical requirements, fixed contracts (e.g. for the government)
  - Agile surgery robot? Agile nuclear power station? Agile Airbag?

- Agility oftenly used for marketing (sales argument)

References and literature

[Beck et al., 2001] Beck et al. (2001): "Manifesto for Agile Software Development", https://agilemanifesto.org/

[Schweizer, 2003] Schweizer, Raffael (2003): „Agile Software Entwicklung mit Scrum“, https://files.ifi.uzh.ch/rerg/amadeus/teaching/seminars/seminar_ws0304/07_Schweitzer_Scrum_Folien.pdf

[Starke, 2002]: Starke, Gernot (2002): „Effektive Softwarearchitekturen“, Hanser-Verlag