

# Three Process Models

**Prof. Dr. Dirk Riehle**

**Friedrich-Alexander University Erlangen-Nürnberg**

**AMOS C02**

Licensed under [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

# Agenda

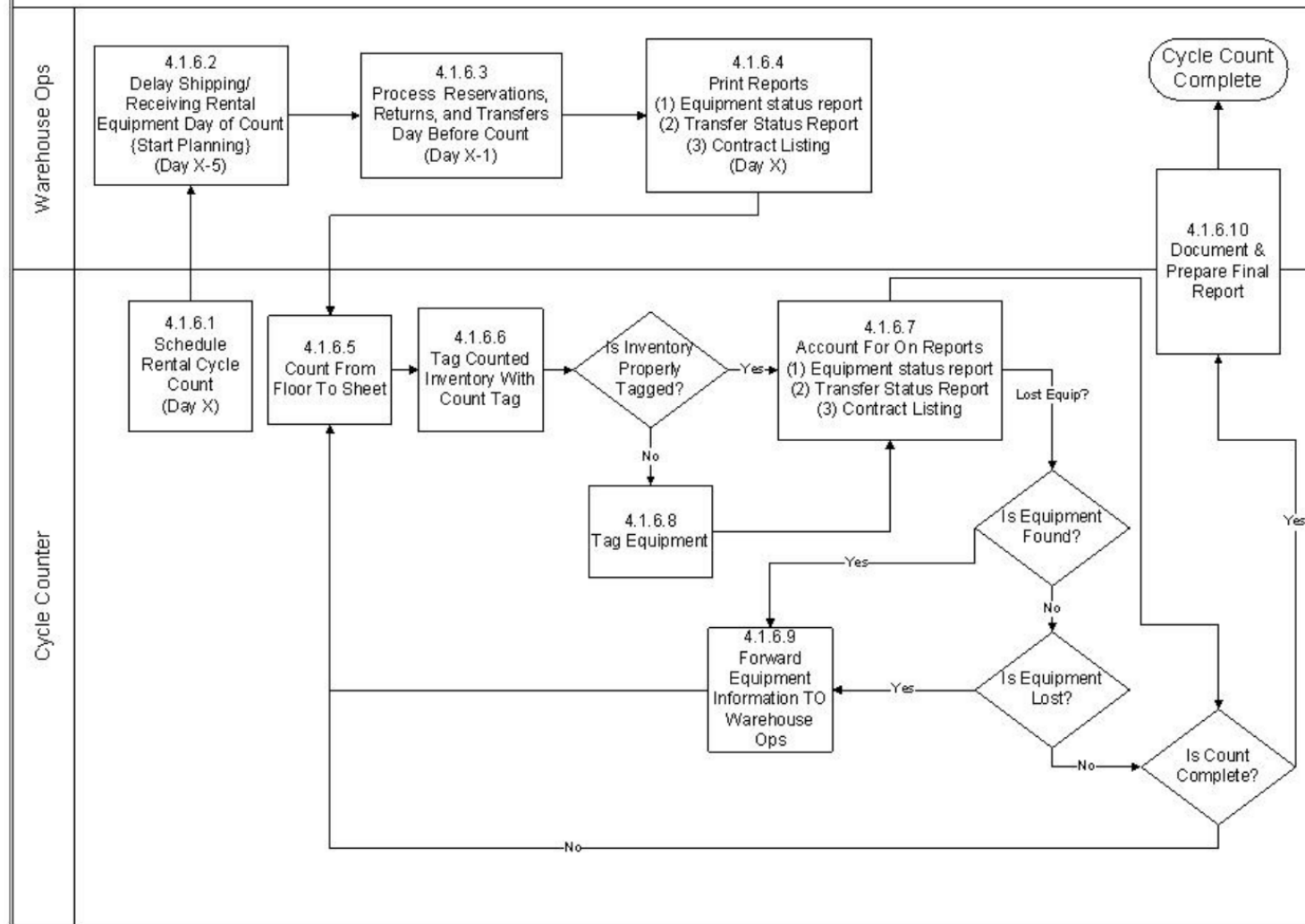
1. Process models
2. Plan-driven development
3. Agile methods
4. Inner source

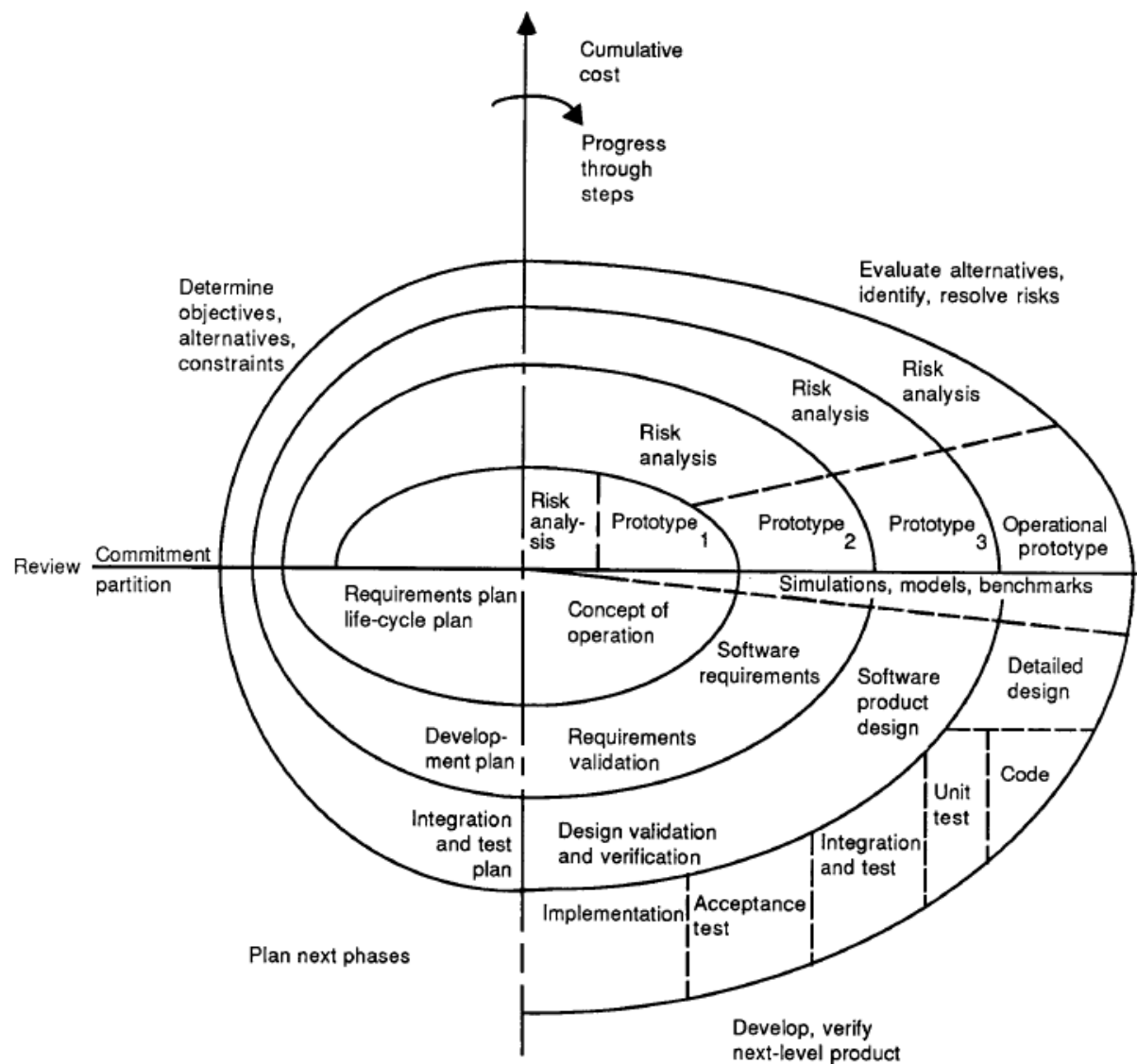
# 1. Process Models

# Software Process Model [DR]

- A **software process**
  - Is a **process** performed with goal of creating and evolving software
- A **software process model**
  - Is a **model** of a **software process**
- Software process **model elements**
  - Comprises **roles, practices, and artifacts** that describe the **valid software process instances**

## 4.1.6 Perform Cycle/Physical Counts (Rental)





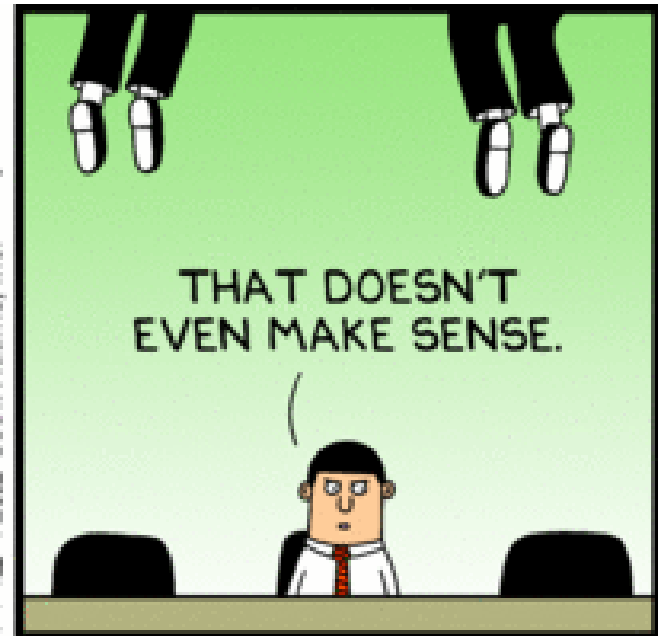
# Project Management Frameworks



Dilbert.com DilbertCartoonist@gmail.com



1-7-12 ©2012 Scott Adams, Inc. Dist. by Universal Uclick

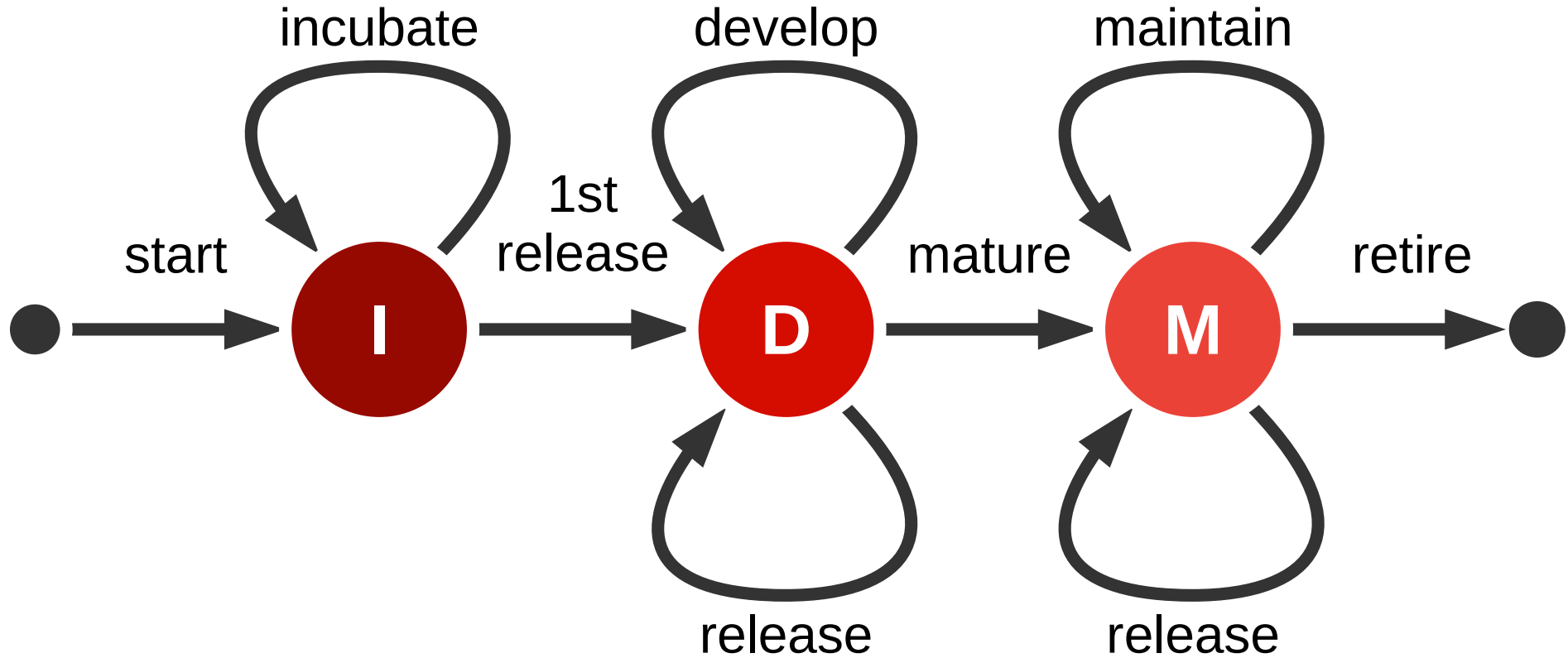


# Key Activities in Software Engineering

- 1. Planning**
- 2. Execution**
- 3. Review**
- 4. Release**

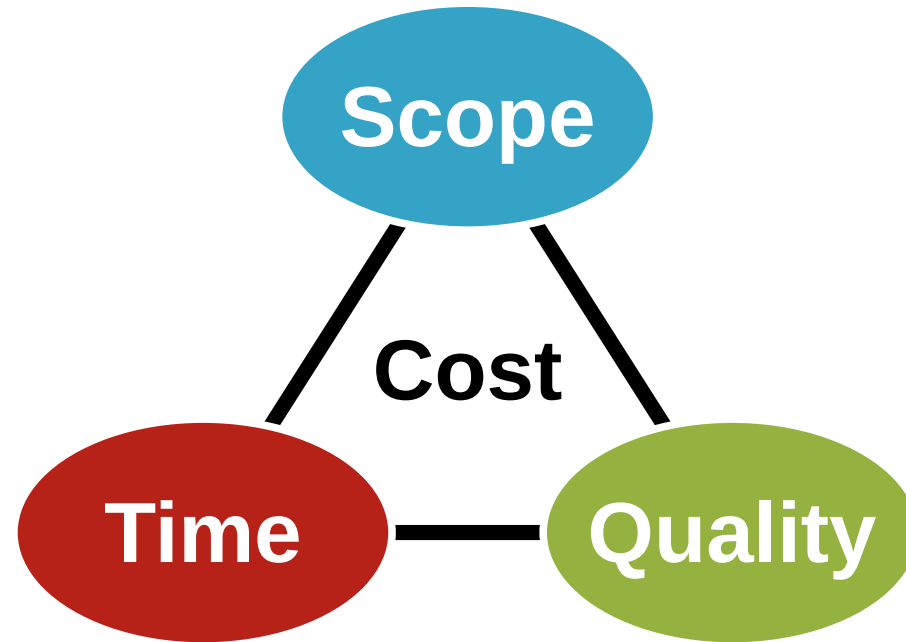


# Example of a Software Product Life-Cycle



I: Incubation  
D: Development  
M: Maintenance

# (One Version of) The Magic Triangle



Cost is usually assumed fixed (defined team).  
Because “adding manpower to a late project makes it later.” [B75]

- 1. Plan-Driven**
- 2. Agile Methods**
- 3. Inner Source**

## **2. Plan-driven Development**

# Plan-Driven Development

- Linear, phase-oriented, software process models
  - Intend to minimize risk through up-front planning
  - Expect only one iteration, start to finish, not many
  - Equate phases with activities
- Examples: Waterfall, V-Modell, RUP



# The Waterfall Model [R04]

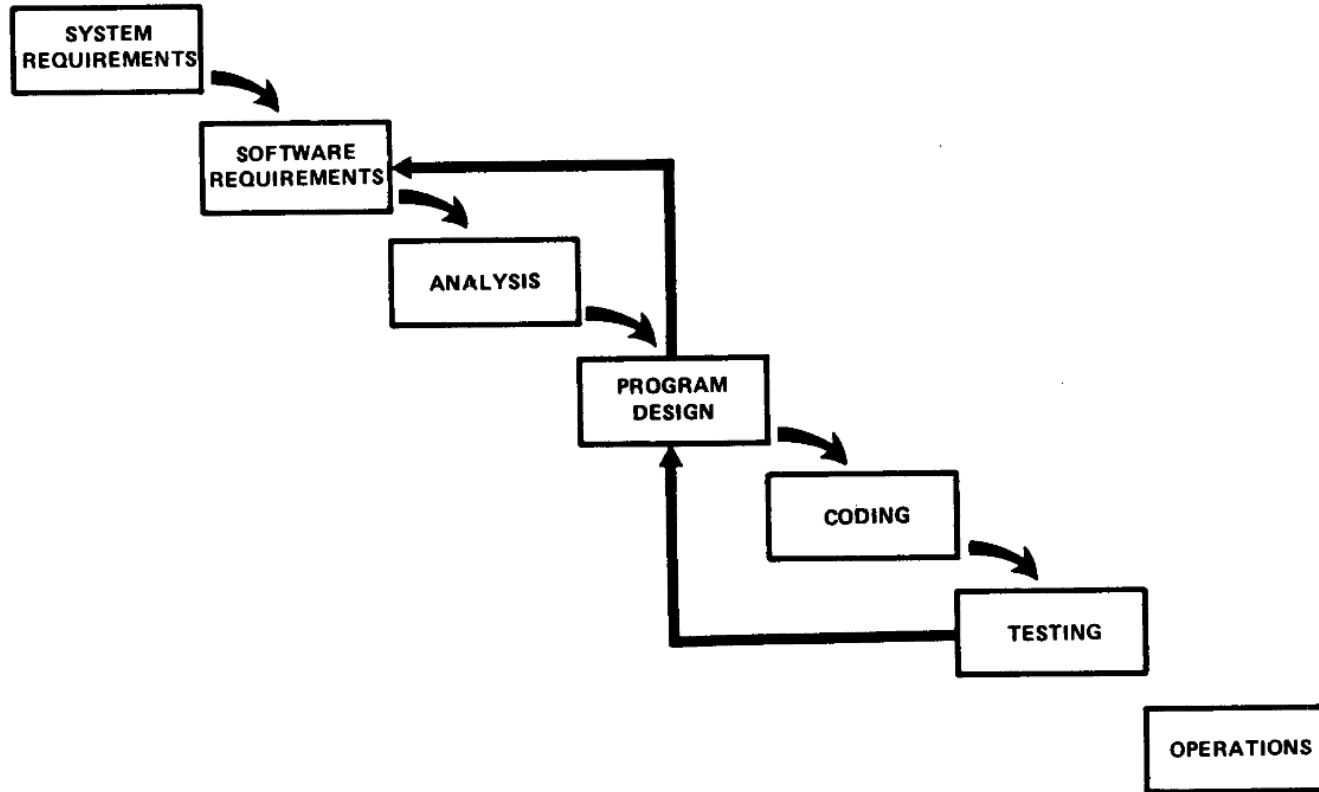


Figure 4. Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.

# Planning in Plan-Driven Development

- Project definition
- Requirements analysis
- System analysis
- Contract negotiation
- ...

# Execution in Plan-Driven Development

- Architecture definition
- System design
- User interface design
- Implementation
- Integration
- ...



# Review in Plan-Driven Development

- System test
- Acceptance test
- Contract validation
- ...

# Release in Plan-Driven Development

- Hand-over
- Deployment
- ...

# Aphorisms on Predicting and Planning

- **“Prediction is very difficult, especially about the future.”**
  - Attributed to Niels Bohr, date unknown
- **“Kein Plan überlebt die erste Feindberührung.”**
  - Helmuth (Karl Bernhard) von Moltke, date unknown
- **“Plans are worthless, but planning is everything.”**
  - Dwight D. Eisenhower, Nov 14, 1957

## Phases $\neq$ Activities

(Activity = performing a practice)

## **3. Agile Methods**

# Video on Predictability of Processes



[1] See <https://youtu.be/D7rbiLNf-JI>

# Video Lesson

- It is impossible to predict such a flight
  - Little bumps on the way have big consequences
  - Little wind gusts will get you way off track
- Executing a plan without steering is dangerous
  - There is no way to ensure you will achieve the desired outcome
  - Belief in flawless execution is, well, flawed and risky
- Also see “the making of Megawoosh” excerpt
  - See [https://youtu.be/\\_n065KE00J0](https://youtu.be/_n065KE00J0)

# Agile Methods

- Invented during the late 1990ties
  - In response to failure of plan-driven methods
  - Driven by consultants as a significant business opportunity
- Repeated iteration over short linear process models
  - Defined equal-length iterations with deliverables
  - Consistent involvement of users for feedback



# Examples of Agile Methods

- Scrum
- XP (eXtreme Programming)
- Adaptive Software Development
- The (set of) Crystal Methods
- Feature-Driven Development
- Pragmatic Programming

# Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

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

Kent Beck  
Mike Beedle  
Arie van Bennekum  
Alistair Cockburn  
Ward Cunningham  
Martin Fowler

James Grenning  
Jim Highsmith  
Andrew Hunt  
Ron Jeffries  
Jon Kern  
Brian Marick

Robert C. Martin  
Steve Mellor  
Ken Schwaber  
Jeff Sutherland  
Dave Thomas

- 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)

# Individuals and Interactions over ...

- **Individuals**

- Trust people
- Allow for self-organization
- Adjust process to people

- **Interactions**

- Get results from collaboration
- Get innovation from people

- **Processes**

- Control people
- Enforce a rigid process
- Adjust people to process

- **Tools**

- Get results from using tools
- Keep people aligned with tools

# Working Software over ...

- **Working Software**

- Get feedback quickly
- Learn from working software
- Steer project from feedback
- Create incremental progress

- **Comprehensive Documentation**

- Wait until the end
- Don't learn at all along the way
- Follow plan until the end
- Delay results until the end

# Customer Collaboration over ...

- **Customer Collaboration**

- Collaborate with customers
- Steer using customer feedback
- Create feedback rhythm
- Allow for change

- **Contract Negotiation**

- Minimize customer contact
- Follow contract-based plan
- Avoid customer feedback
- Stick to agreement

# Responding to Change over ...

- **Responding to Change**

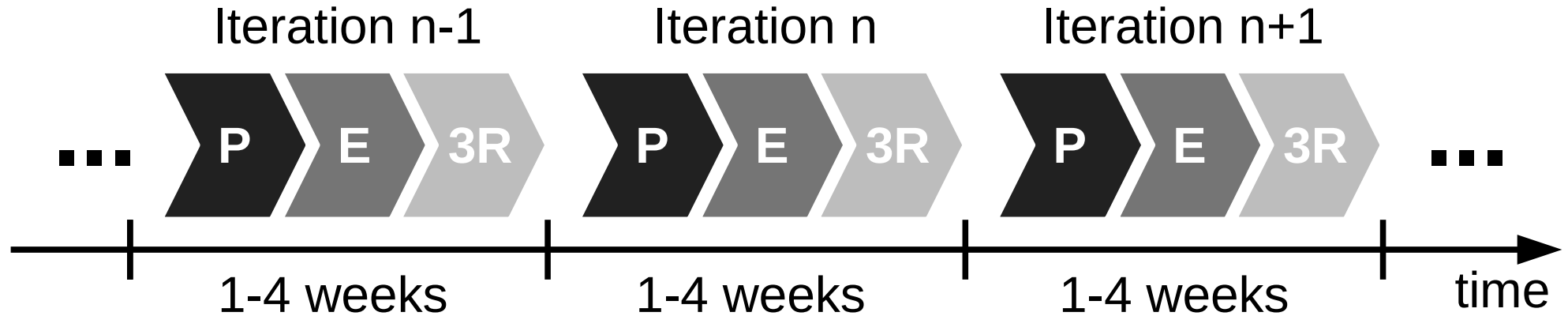
- Adjust to changing reality
- Deliver what customer needs

- **Following a Plan**

- Stick to outdated reality
- Deliver what was negotiated

# Agile Development Process

- Succession of **equal-length iterations** (“time-boxes”)
- Intervention points are during planning and review
- User feedback only available during review



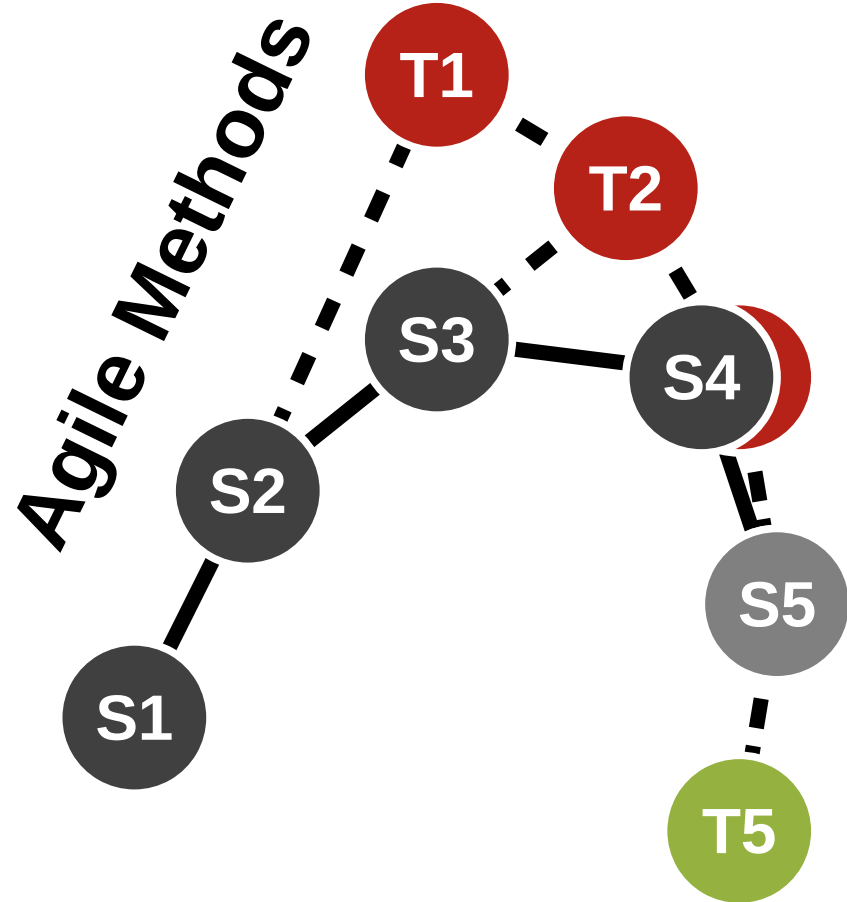
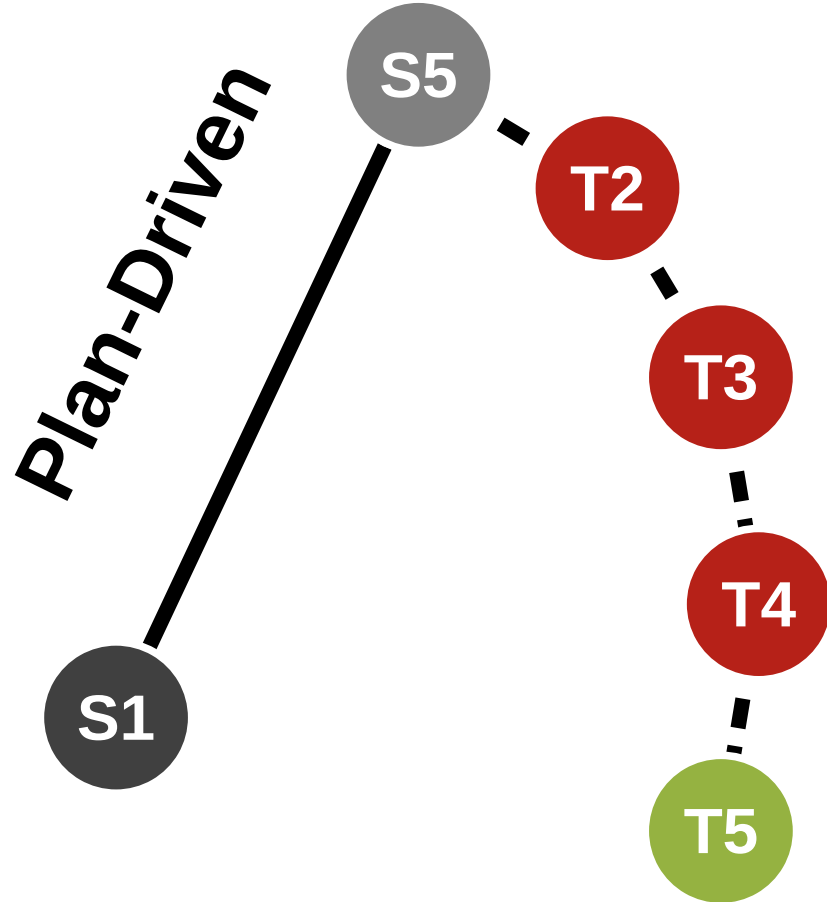
P: Planning  
E: Execution  
3R: Review, release, and retrospective



# Short Iterations and User Feedback

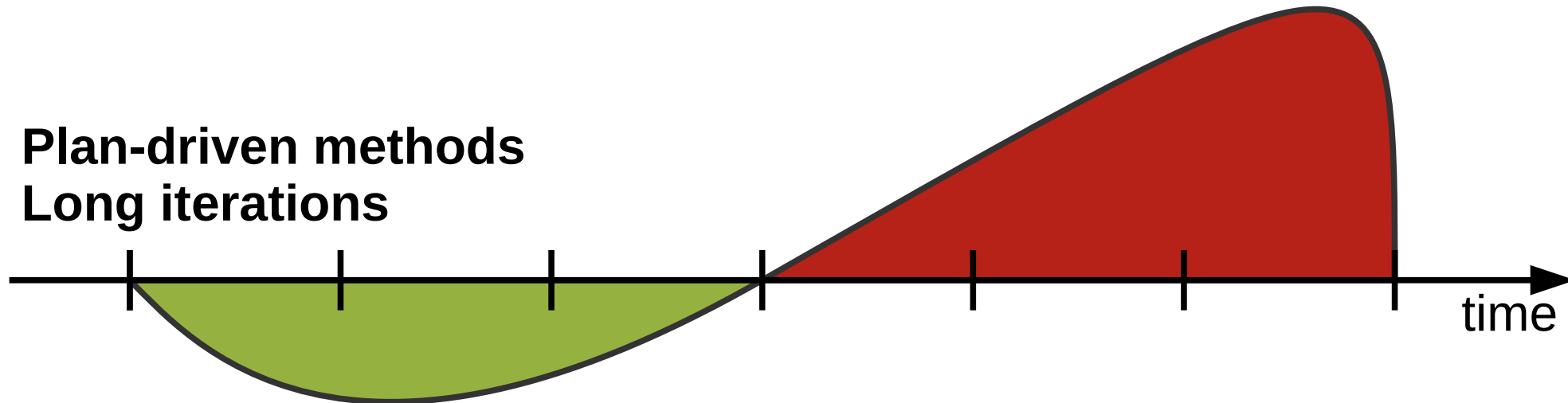
- Short iterations
  - Short iterations lead to focus on high-value features first
  - Established well-worn rhythm is sustainable, avoids burnout
  - Partial functionality is better than none
- User feedback
  - User feedback helps team steer product to meeting needs right
  - Feedback loop ensures that problems surface early
  - Feedback helps recognize and realize new innovative features

# Plan-Driven vs. Agile Processes

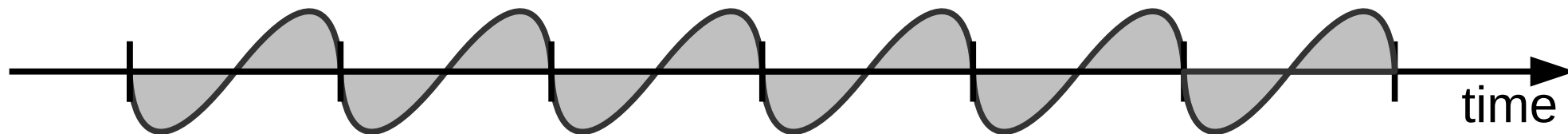


# Plan-Driven vs. Agile Work Rhythms

**Plan-driven methods**  
**Long iterations**



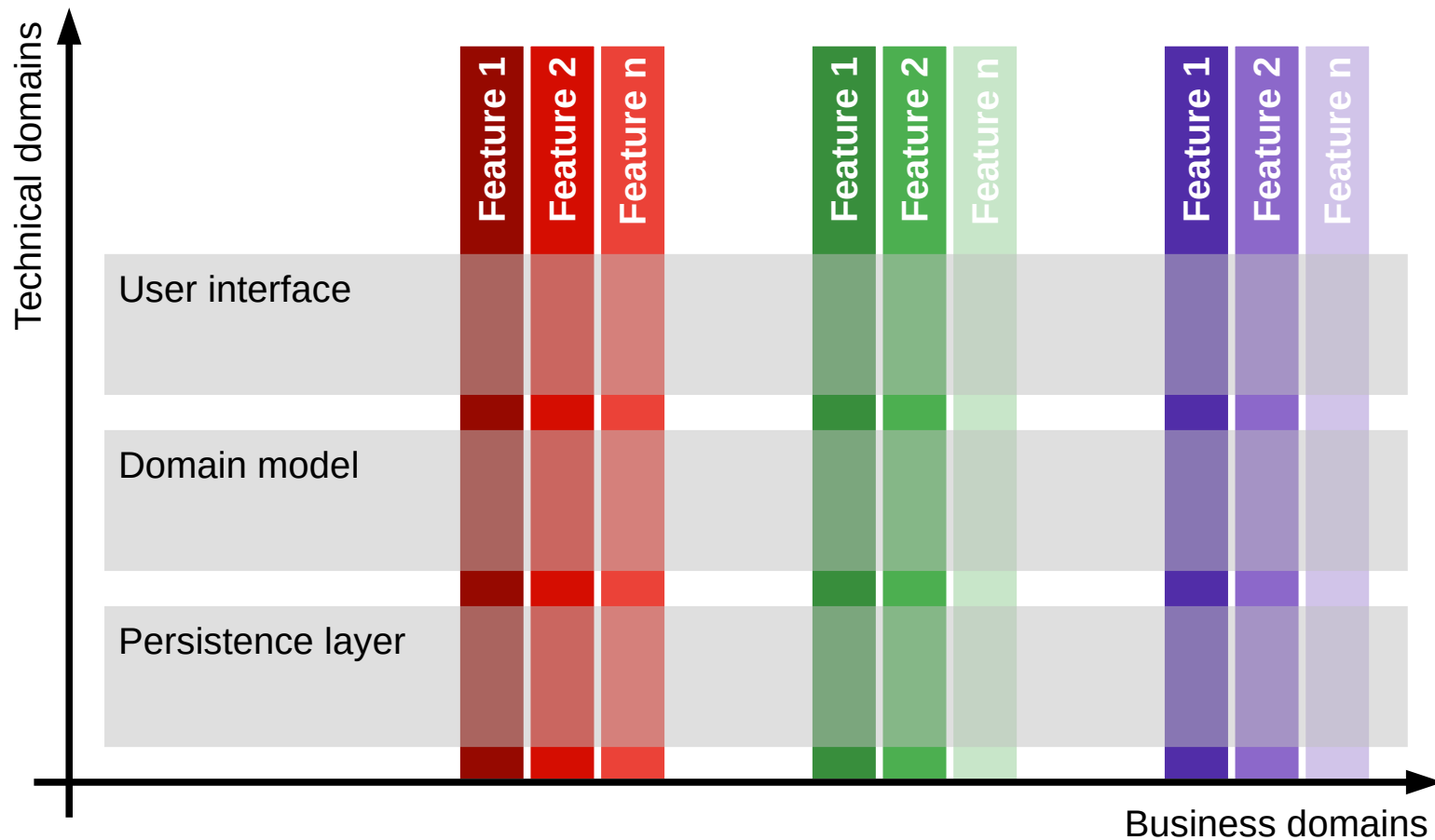
**Agile methods**  
**Short iterations**



**Agile methods  
are high-discipline**

**(more so than plan-driven methods)**

# Challenges to Scaling Agile



## 4. Inner Source

- Inner source software development
  - The use of open source best practices for firm-internal software development

**“Open source** is a development method for software that harnesses the power of **distributed peer review** and **transparency of process**. The promise of open source is **better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in.**” [O12]



- 1. Egalitarian**
- 2. Meritocratic**
- 3. Self-organizing**

# Traditional Work vs. Open Collaboration

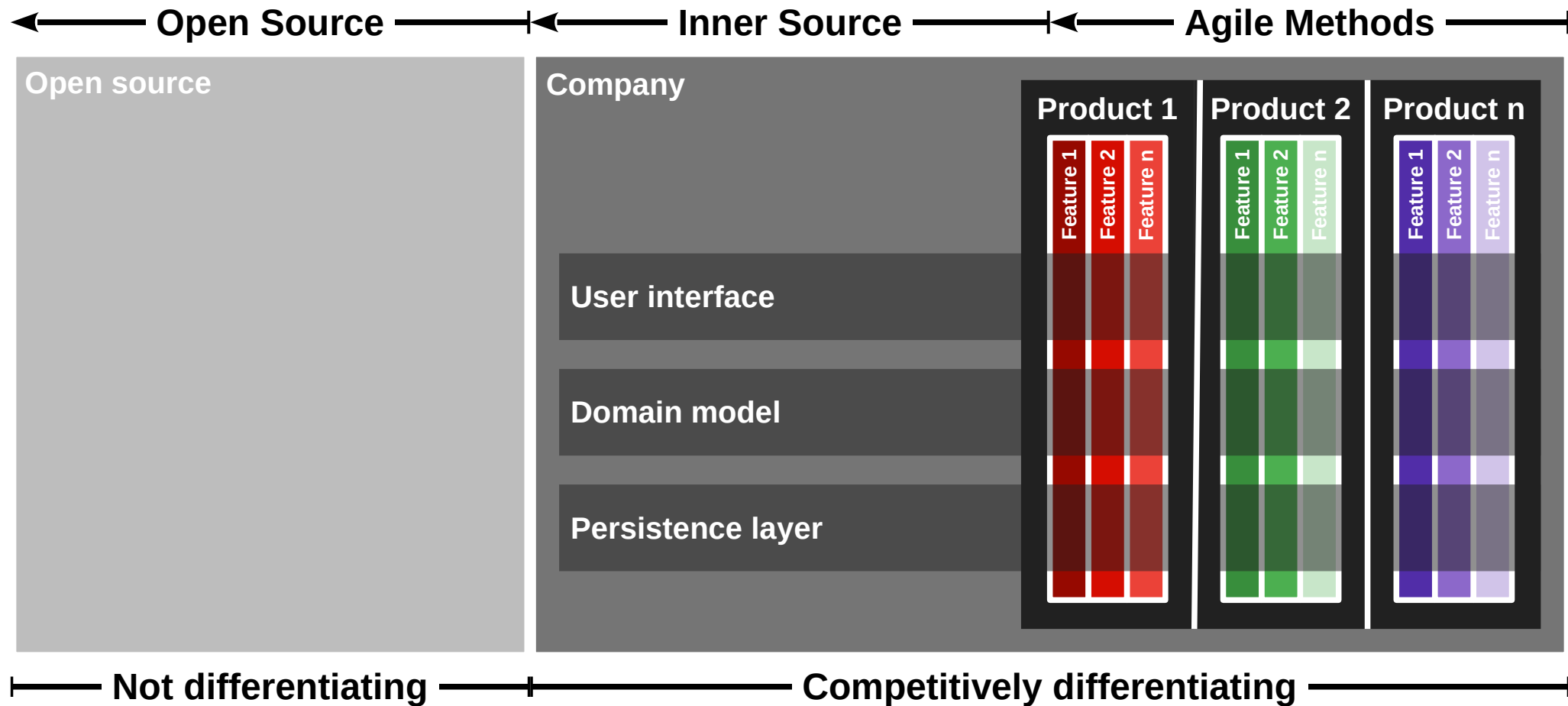
- **Traditional work**

- Hierarchical
  - Closed and hidden silos
  - Assigned to project
- Status-oriented
  - Public + private discussions
  - Hierarchical status decides
- Assigned tasks
  - Prescribed process
  - Prescribed jobs

- **Open collaboration**

- Egalitarian
  - Open for contribution
  - Everyone can contribute
- Meritocratic
  - Public discussion process
  - Decisions based on merit
- Self-organizing
  - People find their project
  - People create their process

# Inner Source vs. Agile Methods



# Comparison of Process Model Types

		Need to Change	
		No	Yes
Need to Scale	No	Plan-Driven Agile Methods Open Source	Plan-Driven <b>Agile Methods</b> Inner Source
	Yes	Agile Methods <b>Plan-Driven</b> Inner Source	<b>Inner Source</b> <b>Open Source</b>

# Quiz on Types of Projects

1. Which process model fits Fixed-Price-Projects?
  - a) Plan-driven process
  - b) Agile methods process
  - c) Open source process
  
2. Which process model fits Time-and-Materials-Projects?
  - a) Plan-driven process
  - b) Agile methods process
  - c) Open source process
  
3. Which process model fits Inter-Firm-Collaboration-Projects?
  - a) Plan-driven process
  - b) Agile methods process
  - c) Open source process

# Review / Summary of Session

- Key activities in software engineering
- Main categories of process models
  - Plan-driven methods
  - Agile methods
  - Inner source
- Plan-driven vs. agile methods
  - Predictability of the future
  - The agile manifesto

# Thank you! Questions?

[dirk.riehle@fau.de](mailto:dirk.riehle@fau.de) – <https://oss.cs.fau.de>

[dirk@riehle.org](mailto:dirk@riehle.org) – <https://dirkriehle.com> – [@dirkriehle](#)

# License Declaration and Copyright Notices

- License
  - Licensed under the [CC BY 4.0 International](#) license
- Copyright
  - © 2021 Dirk Riehle, some rights reserved