Agile Software Development in Corporate Environments

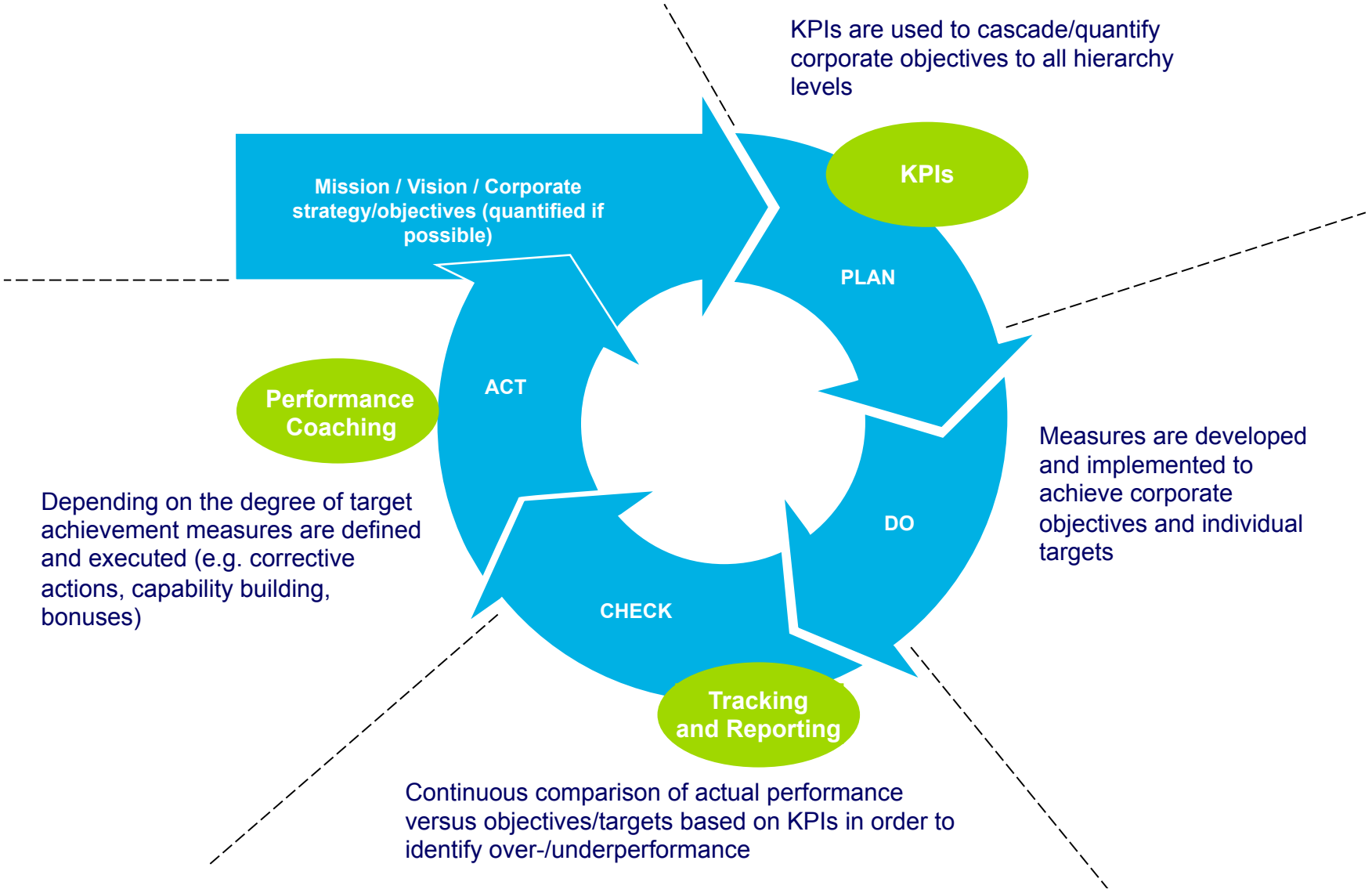
Dr. Alexander Schatten
alexander@schatten.info
http://www.schatten.info

Overview

- 20 years (70 years?) agile practices
- Motivation
- The *Agile Manifesto*
- eXtreme Programming, SCRUM, (Software) Kanban
- Corporate challenges
- Requirements and transparency
- Balancing Measurement and Self-Organisation

20 YEARS AGILE PRACTICES

Plan / Do / Study / Act (1940)



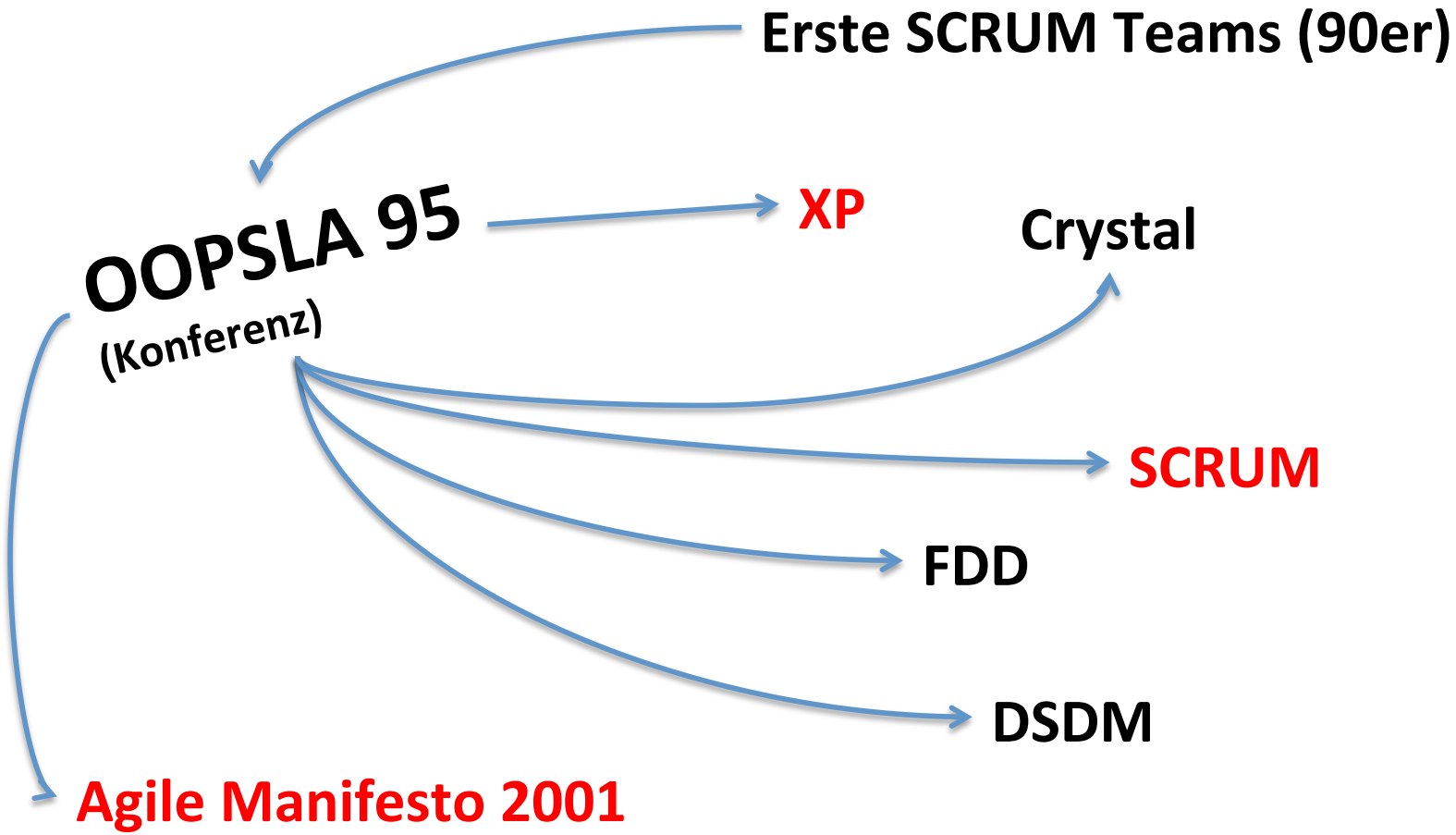
Motivation

- Organisational challenges
- Technical challenges
- The illusion of predictability and forecast reliability

- *… and also: we are just doing what every other industry does: approach the product step by step before (mass) production*

Process	Waterfall Development	Iterative and Incremental	Agile Development
Measure of Success	Conformance to plan →		Response to change, working code
Management Culture	Command and control →		Leadership/ collaborative
Requirements and Design	Big and up front →		Continuous/emergent/ just-in-time
Coding and Implementation	Code all features in parallel/test later →		Code and unit test, deliver serially
Test and Quality Assurance	Big, planned/ test late →		Continuous/concurrent/ test early
Planning and Scheduling	PERT/detailed/fix scope, estimate time and resource →		Two-level plan/fix date, estimate scope

The First 20 Years…



Software Kanban (~2007)

Agile Manifesto

1. Individuals and Interaction
   over Processes and Tools
2. Working Software
   over Comprehensive Documentation
3. Customer Collaboration
   over Contract Negotiation
4. Responding to Change
   over Following a Plan

Agile Practices Today

- Prozess-oriented
  - SCRUM
  - Software Kanban
- Methodical building blocks
  - eXtreme Programming
- Geeks only?
  - Meanwhile ~15 years of experience of agile methods in corporate environments
  - Usage in small and large international companies, but also in Austria (e.g. banks, insurance companies, …)
  - Usage in small projects (individual teams, 3–10 devs)
  - Usage in large projects (multiple teams, hundreds of devs)

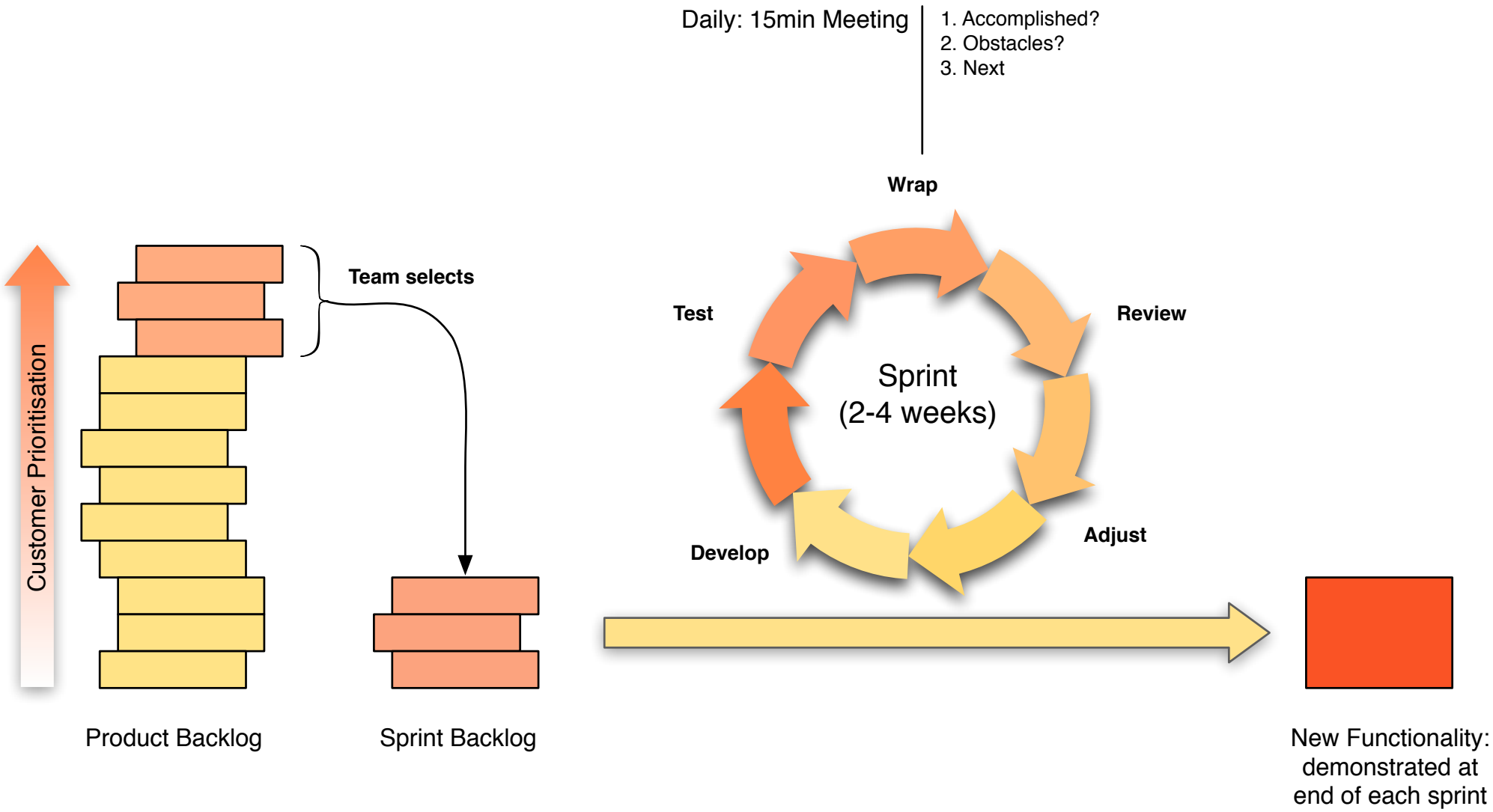
EXTREME PROGRAMMING

XP-Praktiken

- **Communication / Collaboration / Architecture**
  - Planning Game (WS)
  - Metaphor (WS)
  - Simple Design (WS)
- **Process**
  - Small Releases
  - Pair Programming
  - Collective Code Ownership (WS)
  - 40-hrs Week
  - On-Site Customer (WS)
- **Technical**
  - Coding Standards
  - Testing
  - Continuous Integration
  - Refactoring

SCRUM

SCRUM Prozess

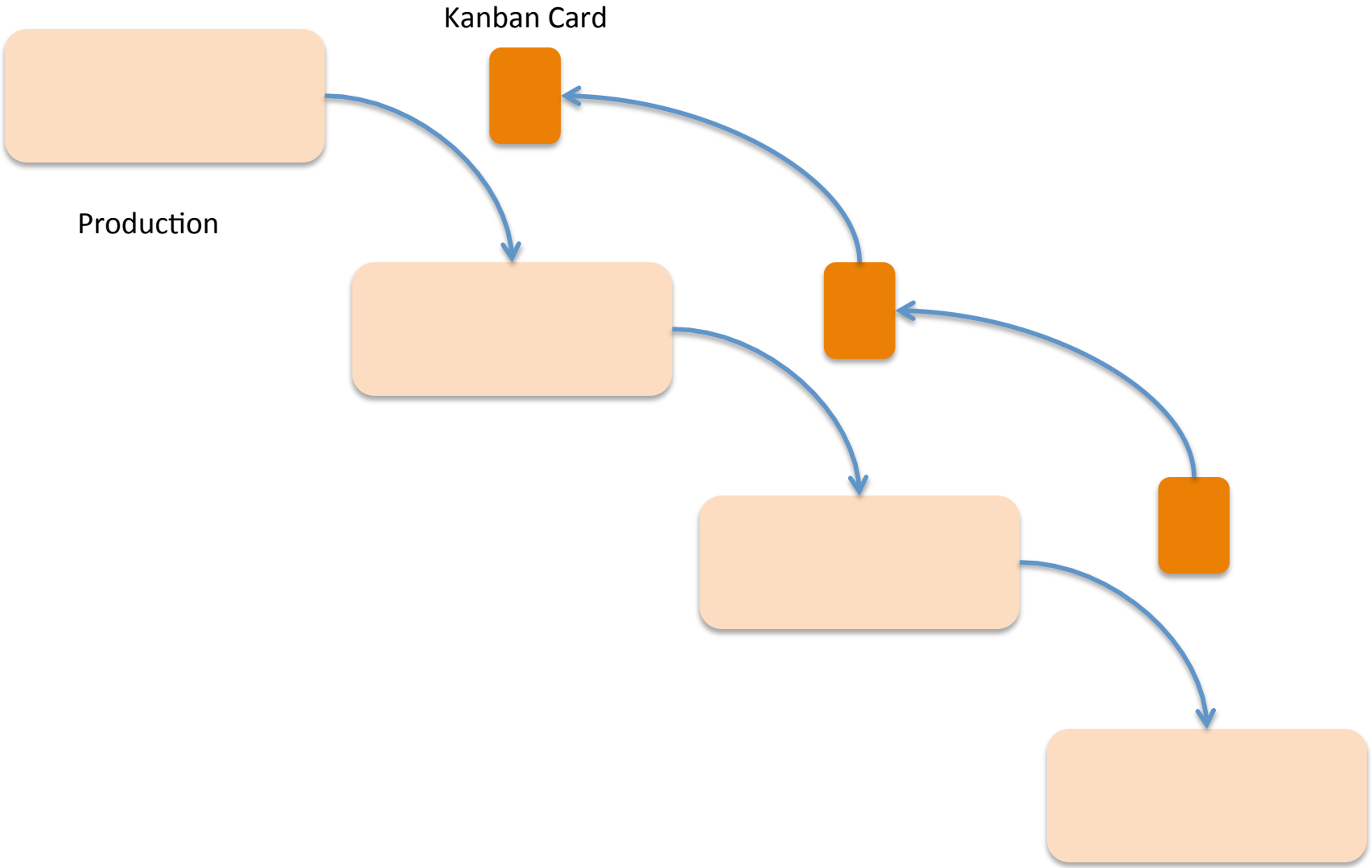


SCRUM Prozess

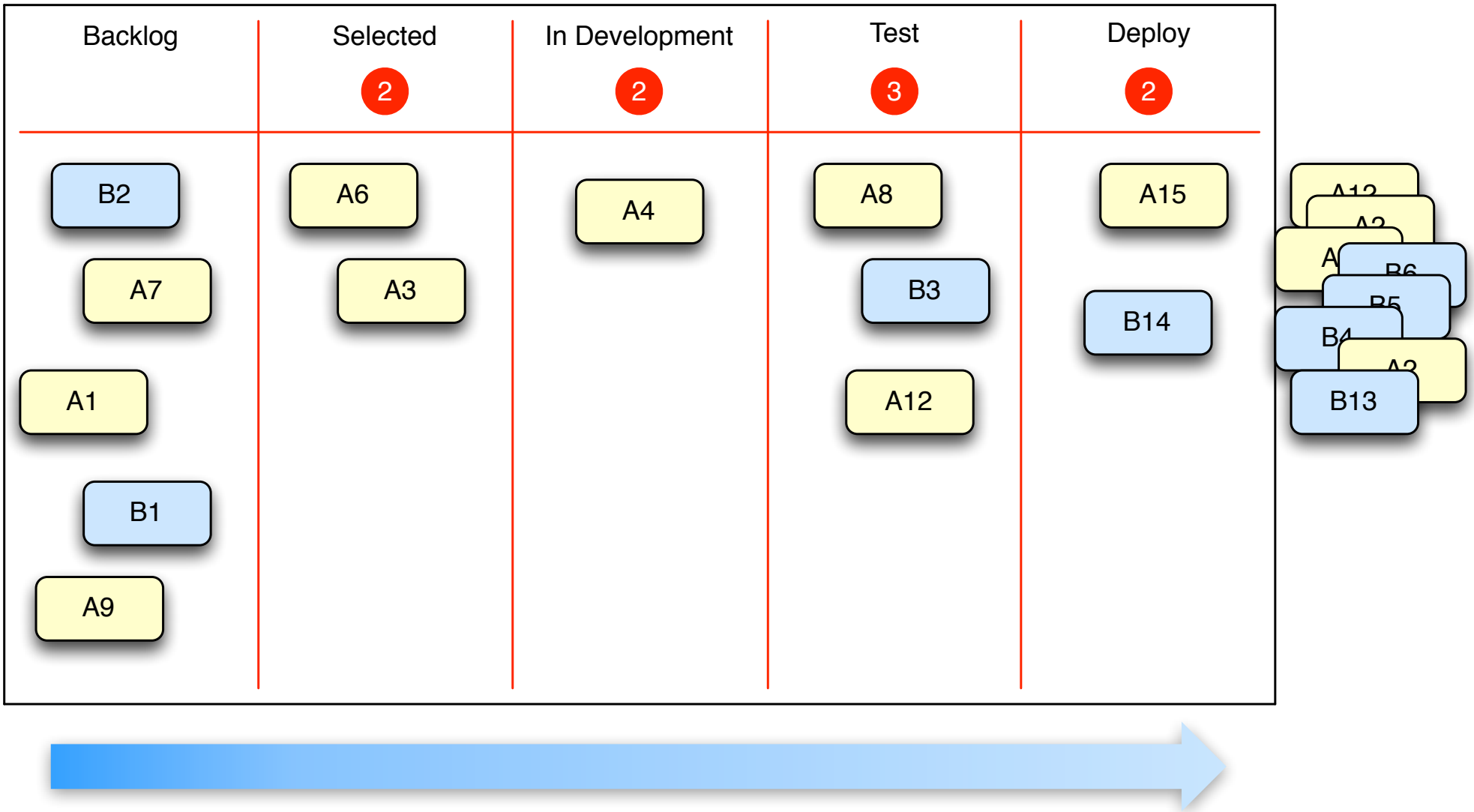
- Work in **Iterations**
- **Team size** < 10 Personen
- Customer is tightly integrated in the process
- Realistic **estimations** hand in hand with **controlling**
  - User Stories
  - Backlogs
  - “Planning Poker”
- **Team-estimations** and **Performance** (checked at each iteration)
- Undisturbed and focussed work for teams during iterations

SOFTWARE KANBAN

“Stop Starting, Start Finishing…”



Workflow Visualisation



Software Kanban

- Focus on “flow”, avoiding bottlenecks
- Visualisation of current process flow and activities
- Real-time metrics (KPIs), e.g.
  - Average Lead Time
  - Cumulative Flow Diagrams: Cycle Time

AGILE PRACTICES IN CORPORATE ENVIRONMENTS

Agile Practices? Startups Only?

- Salesforce
- Google
- Yahoo
- Intel
- Siemens
- McKinsey
- Philips
- JP Morgan
- Bank Austria, BAWAG PSK, Raiffeisen
- Fabasoft, Frequentis, Tricentis, Anecon, Accenture, Kapsch, VIG
- Usw.

Challenges in Corporate Environments

Some challenges to expect 1/2

- Multiple Teams
- Many developers
- Large projects versus perfective maintenance
- Priorisation (very different goals among stakeholders: business teams, management, IT)
- Projects partly internal and partly external (fixed price, …)
- Sceptical customers/business experts (from failures in the past, *"most IT procjects fail…"; fear of new processes, efficiency measures, etc.*)
- Management levels & involvement
- Budgeting and planning cycles
- Long and complicated projects, e.g. replacement of legacy systems
- Lack of know how (from domain side and technical side alike), partly due to long runtime of legacy systems
- Reporting and controlling (different expectations: financial, progress, quality, …)
- Diffusion of responsibility in large organisations

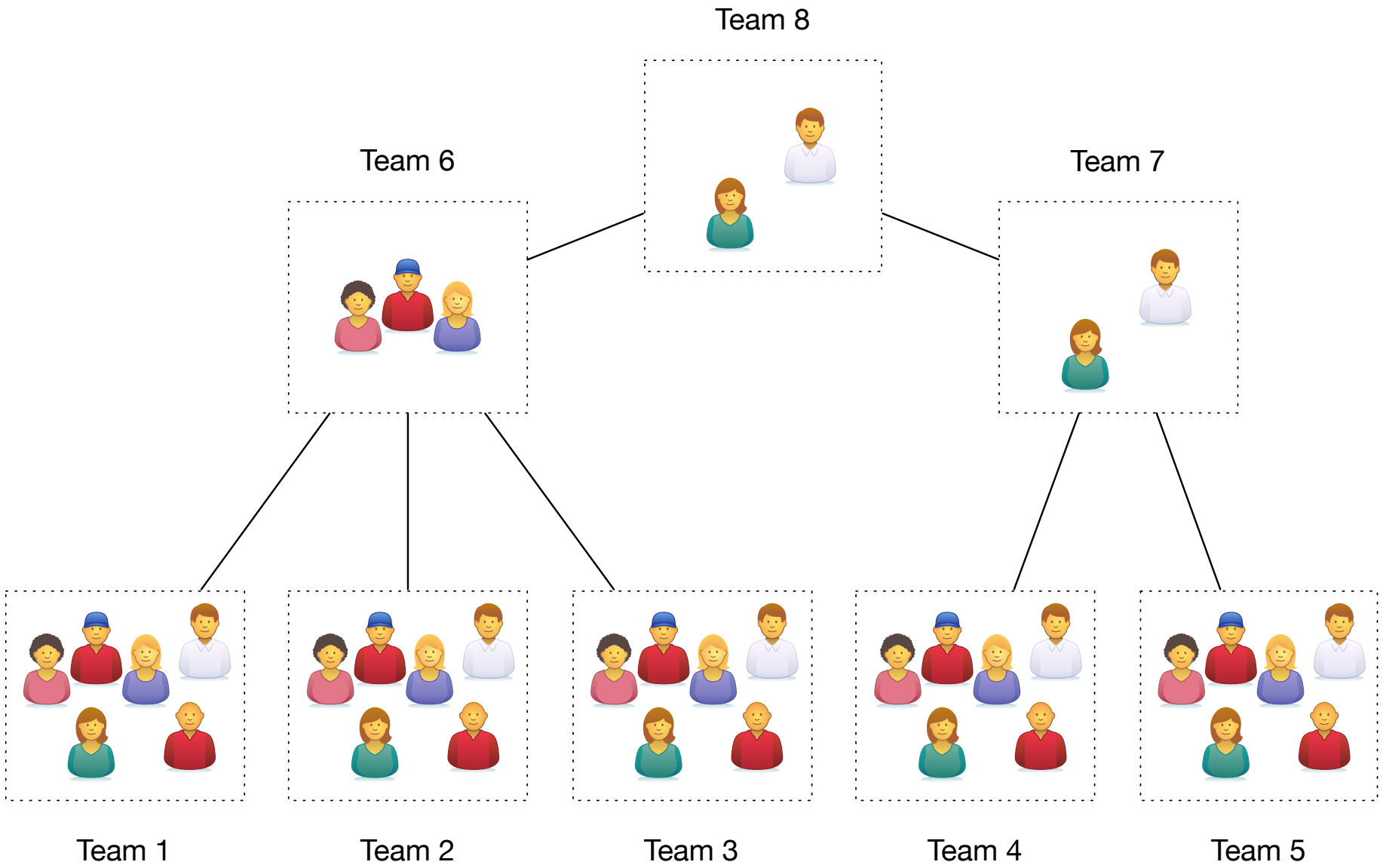
Some challenges to expect 2/2

- What is the core of agile software development?
  - Flexible self-organisation of teams around product owner
  - Lean and efficient work in small teams and short iteration cycles
- What is the core demand of programs and managers of large interconnected systems?
  - The opposite

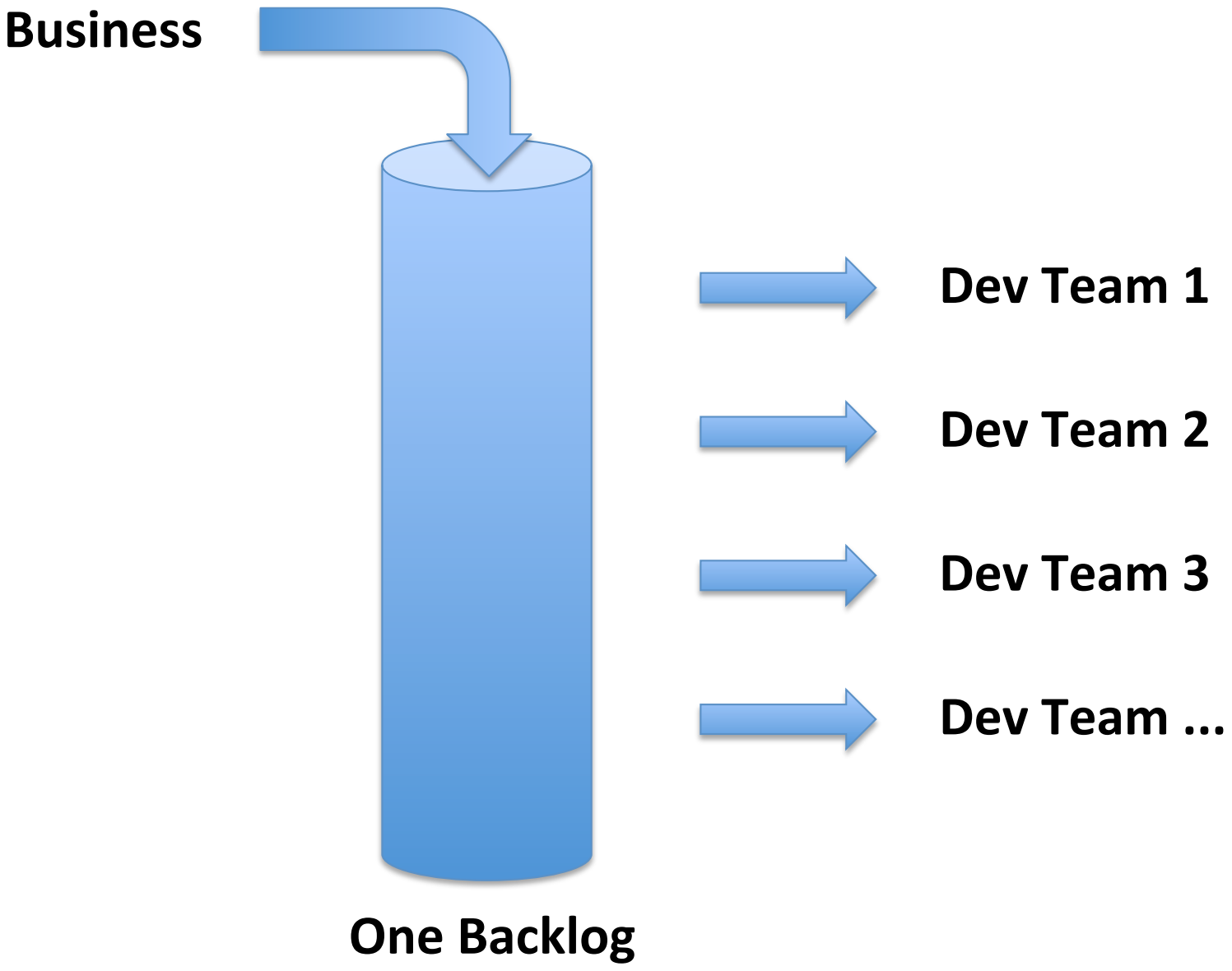
Side Remark

- *Knowing something does not work seldom discourages people from still doing it, e.g.*
  - *Bonus schemes*
  - *Teamwork*
  - *Long-term planning and budgeting*

Scrum of Scrums

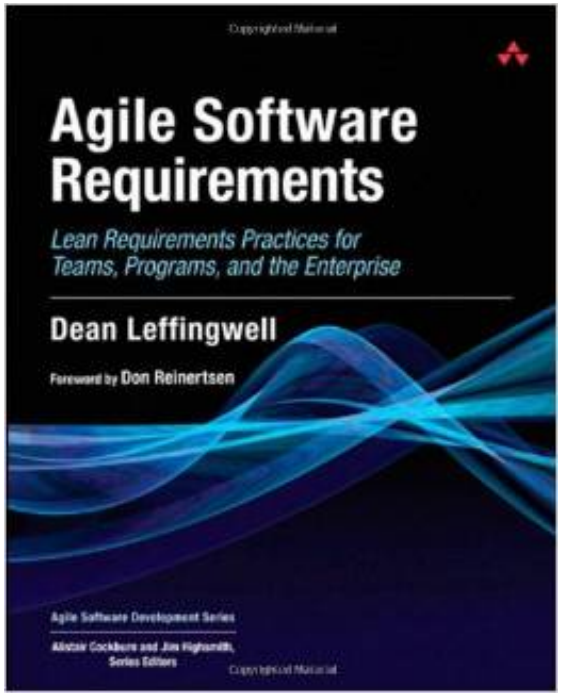


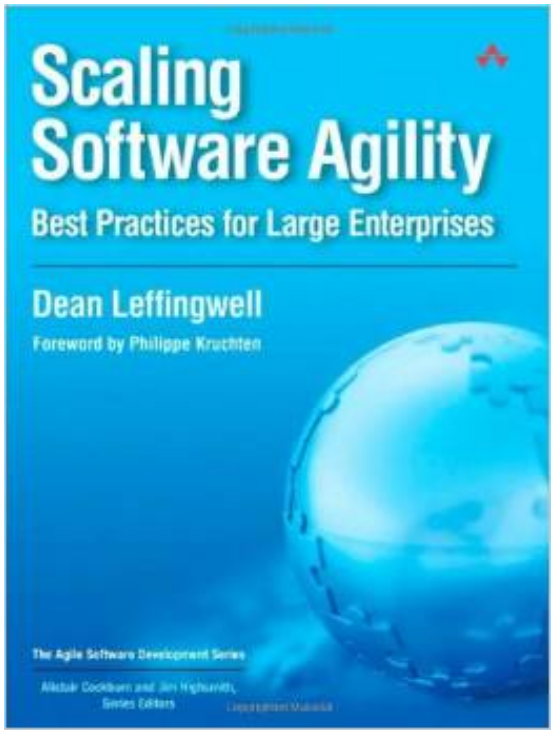
“Factory” Approach

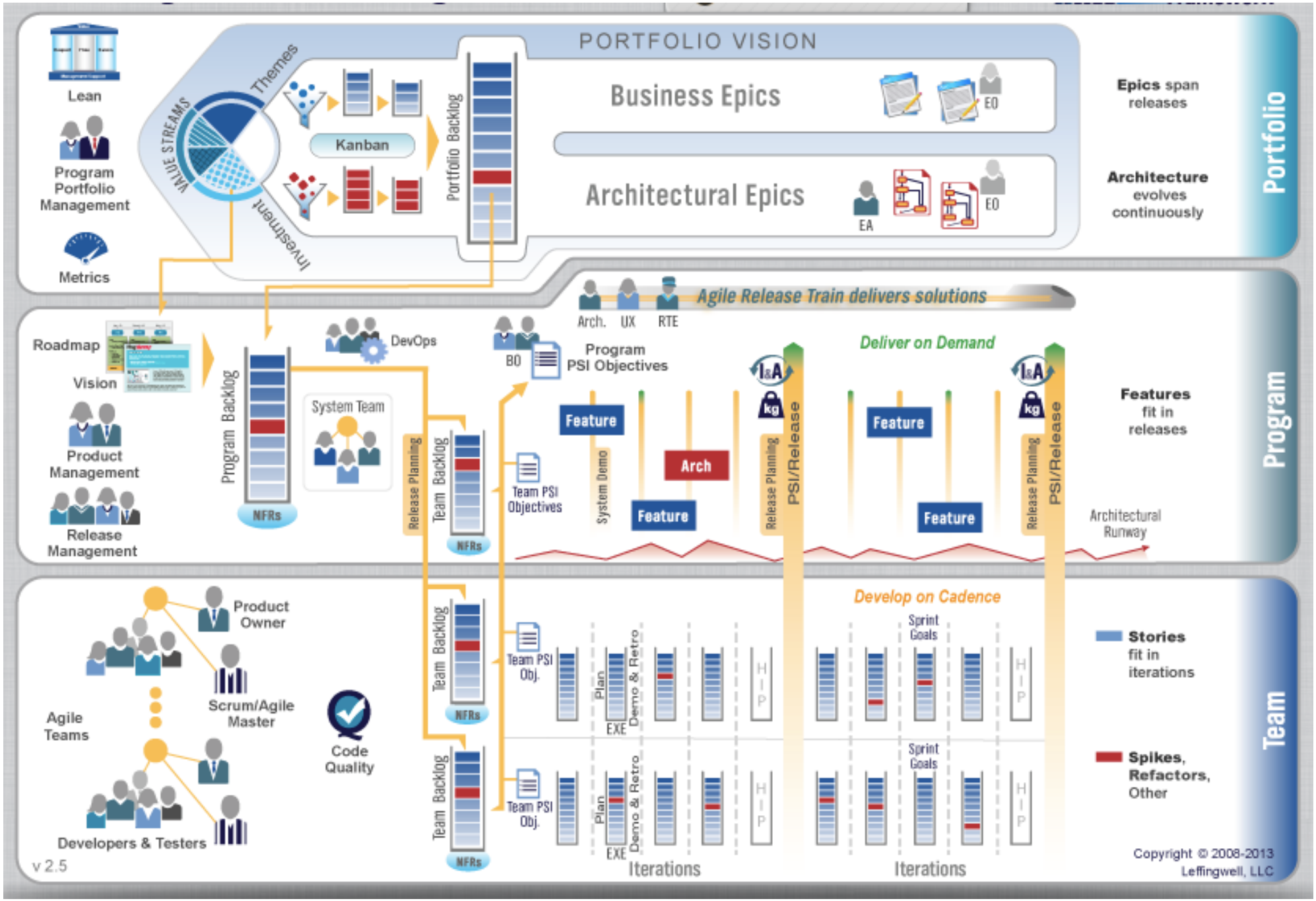


SAFE Framework

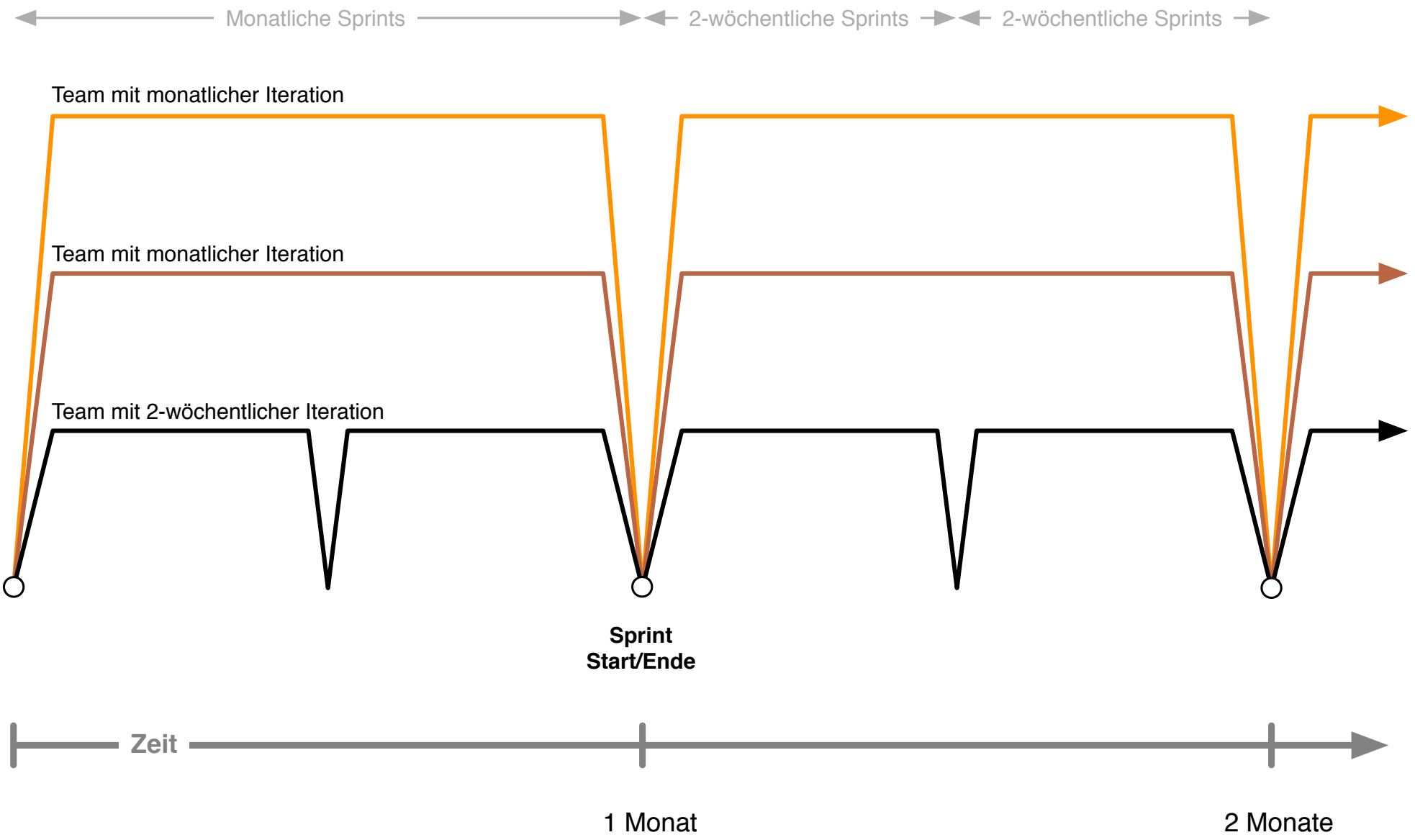
Dean Leffingwell: Scaled Agile Framework
http://scaledagileframework.com



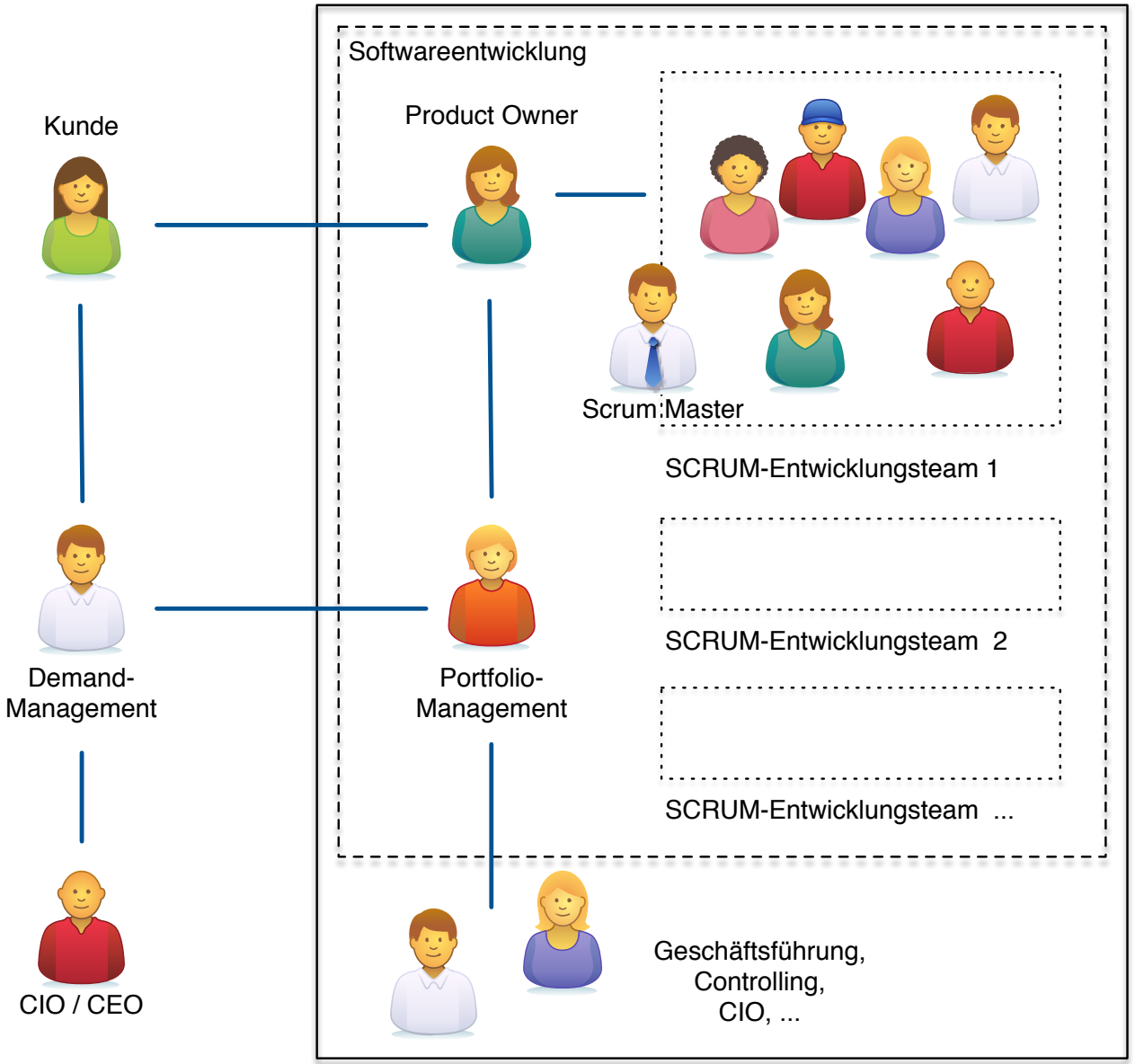



PORTFOLIO VISION
Lean
Program Portfolio Management
Metrics
VALUE STREAMS
Themes
Investment
Kanban
Portfolio Backlog
Business Epics
EO
Architectural Epics
EA
EO
Epics span releases
Architecture evolves continuously
Portfolio
Agile Release Train delivers solutions
Arch.
UX
RTE
Roadmap
Vision
Product Management
Release Management
Program Backlog
NFRs
DevOps
System Team
BO
Program PSI Objectives
Team PSI Objectives
Release Planning
Team Backlog
NFRs
Deliver on Demand
Feature
System Demo
Arch
Feature
Feature
Feature
I&A
kg
Release Planning
PSI/Release
Architectural Runway
Features fit in releases
Program
Product Owner
Scrum/Agile Master
Agile Teams
Developers & Testers
Code Quality
v 2.5
Team Backlog
NFRs
Team PSI Obj.
Develop on Cadence
Plan
Demo & Retro
EXE
H-P
Sprint Goals
Iterations
Iterations
Stories fit in iterations
Spikes, Refactors, Other
Team
Copyright © 2008-2013 Leffingwell, LLC

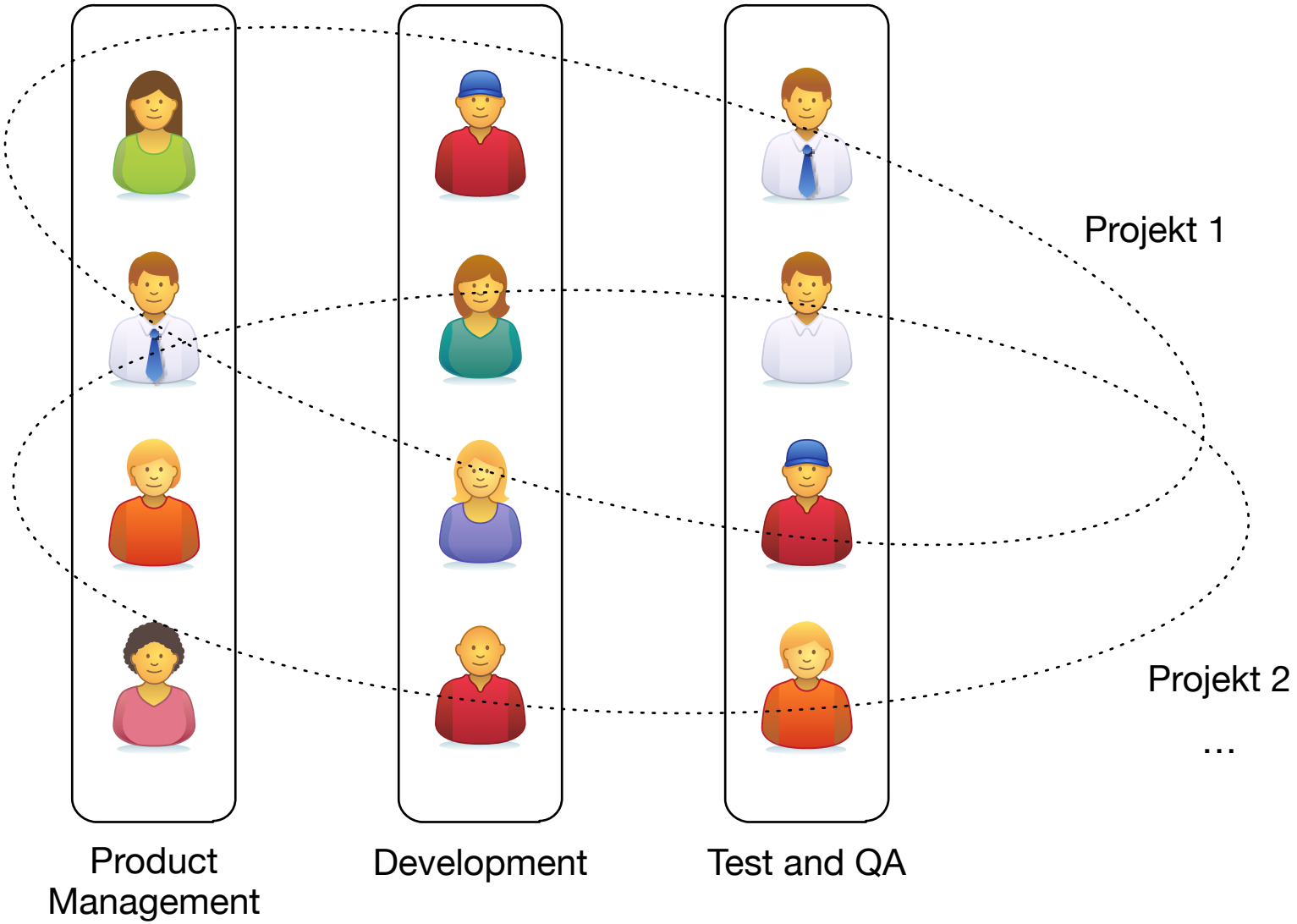
Portfolio Approach



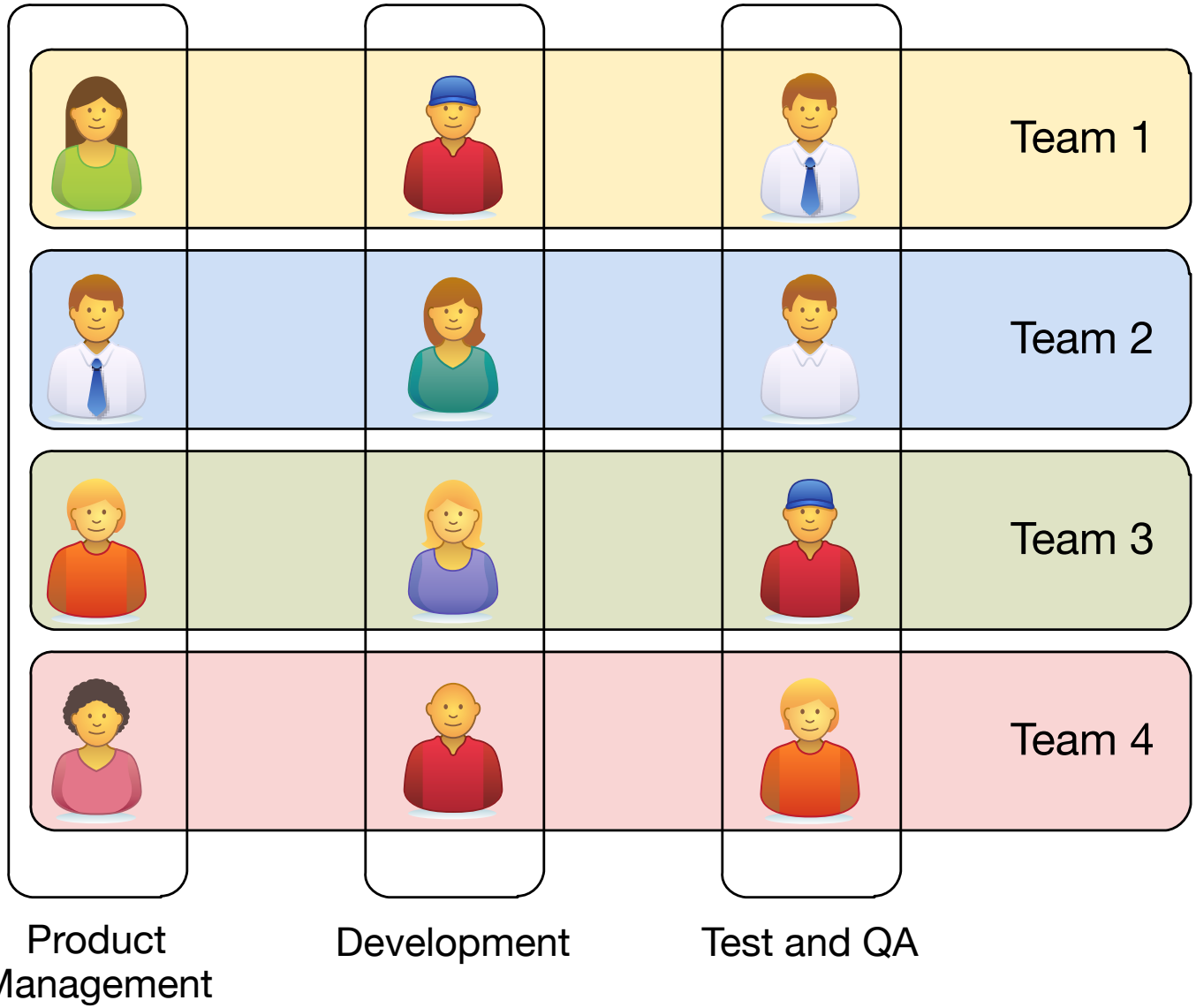
Roles in Agile Processes



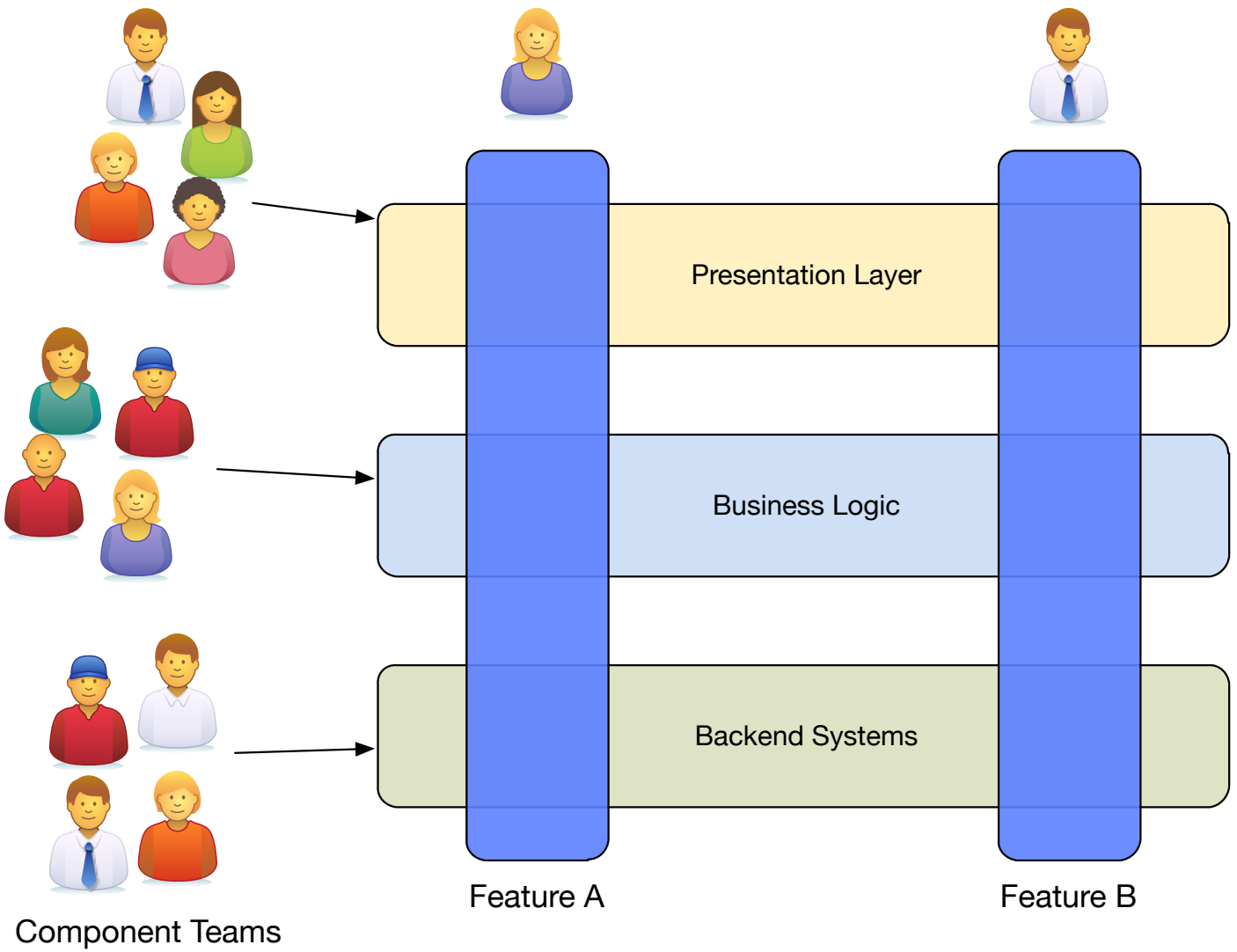
Functional Silos (traditional)



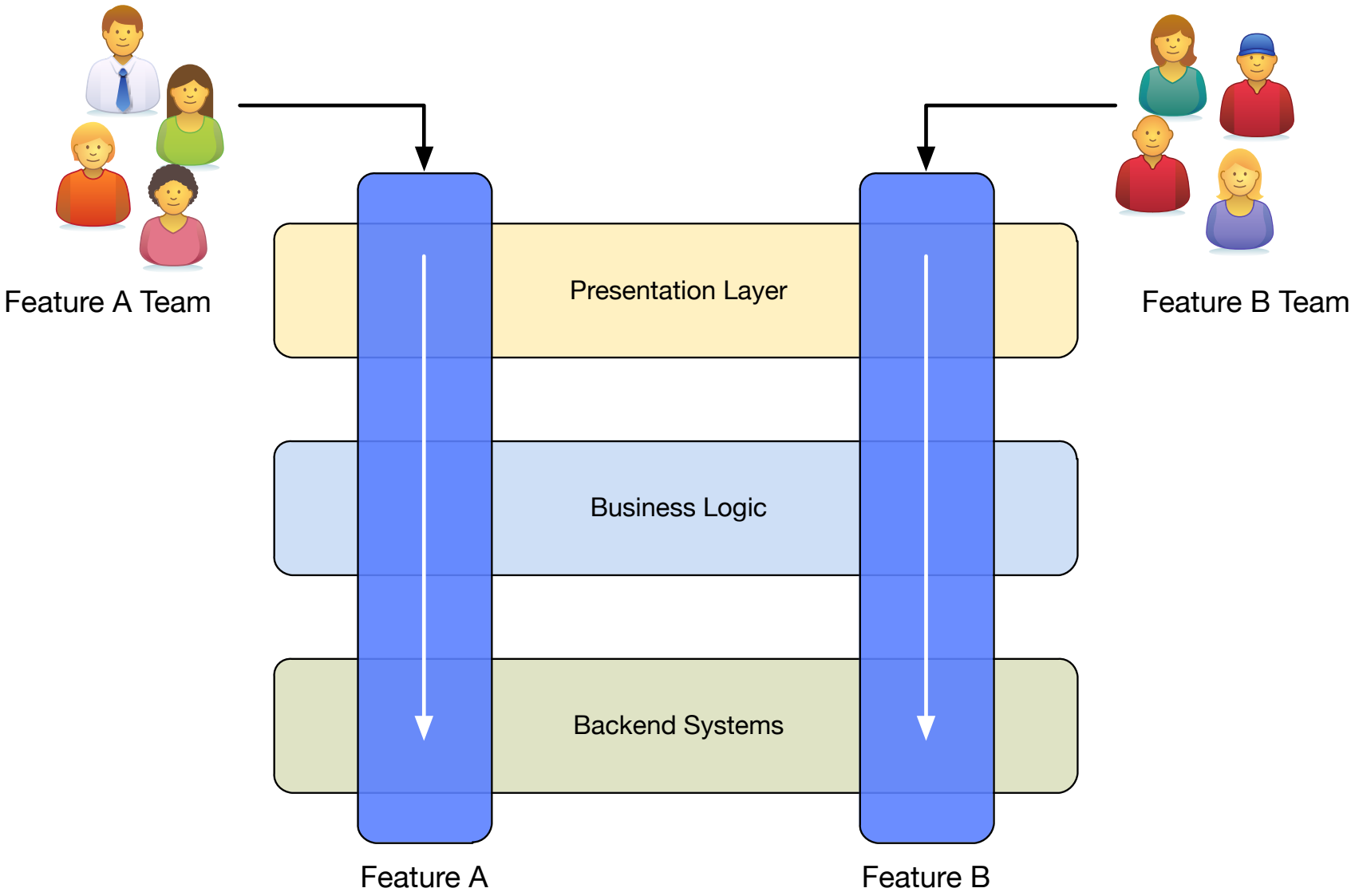
Agile Teams


Team 1
Team 2
Team 3
Team 4
Product Management
Development
Test and QA

Organisation Following Components (traditional)



Agile Team Organisation
Following Features / Processes / Services



Remark

- *Definition of a service is much more complicated than often assumed*
- *One service / feature often invokes many (complicated and complex) components*
- *Very difficult to cut agile teams, particularly with legacy systems involved*

Customer Responsibility

- Product Owner
- Roles and process responsibilities have to be clarified
- Lack of clear roles and responsibilities is often the main factor for failure in agile projects!

Further Roles

- SCRUM-Master
- System Team
- Product Manager
- Operations

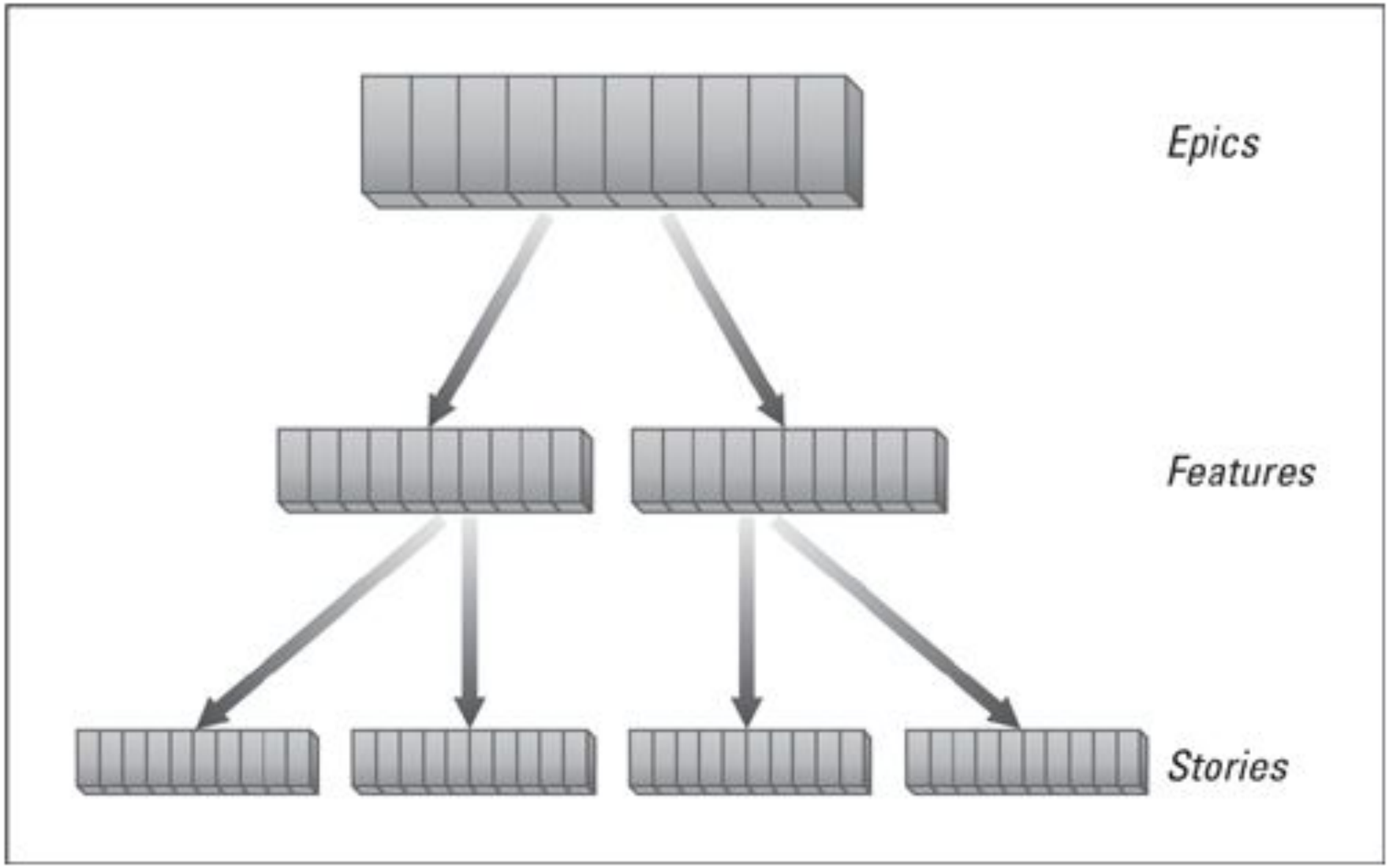
- *Remark: there is usually a "natural conflict" between software development teams and operations team; as well as discussions on the actual role of operations (see DevOps trend)*

Contracts and Accounting

- Fixed price project
- Cooperation model
  ("Time and Material")

REQUIREMENTS, QS, TRANSPARENCY

Requirement Engineering


Epics
Features
Stories

Type of Information	Description	Responsibility	Time Frame and Sizing	Expression Format	Testable
Investment theme	*Big*, audacious, game changing, initiatives. Differentiating, and providing competitive advantage.	Business executives, Portfolio management.	Span strategic planning horizon, 12 to 18+ months. Not sized, controlled by percentage investment.	Any: text, prototype, PPT, video, conversation.	No
Epic	Bold, impactful, marketable differentiators.	Portfolio management. Business analysts, product and solution management, system architects.	6 to 12 months. Sized in points.	Most any, including prototype, mock-up, short phrase, or vision statement.	No
Feature	Short, descriptive, value delivery and benefit-oriented statement. Customer and marketing understandable.	Product manager and product owner.	Fits in an internal release (PSI), divide into incremental subfeatures as necessary. Sized in points.	Key phrase or user story voice form. May be elaborated with system use cases.	Yes
Story	Small, atomic. Fit for team and detailed user understanding.	Product owner and team.	Fits in a single iteration. Sized in story points.	User story canonical form.	Yes

User Stories

As WHO I want WHAT so that WHY

Acceptance Criteria

Small, “one card”

INVEST

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

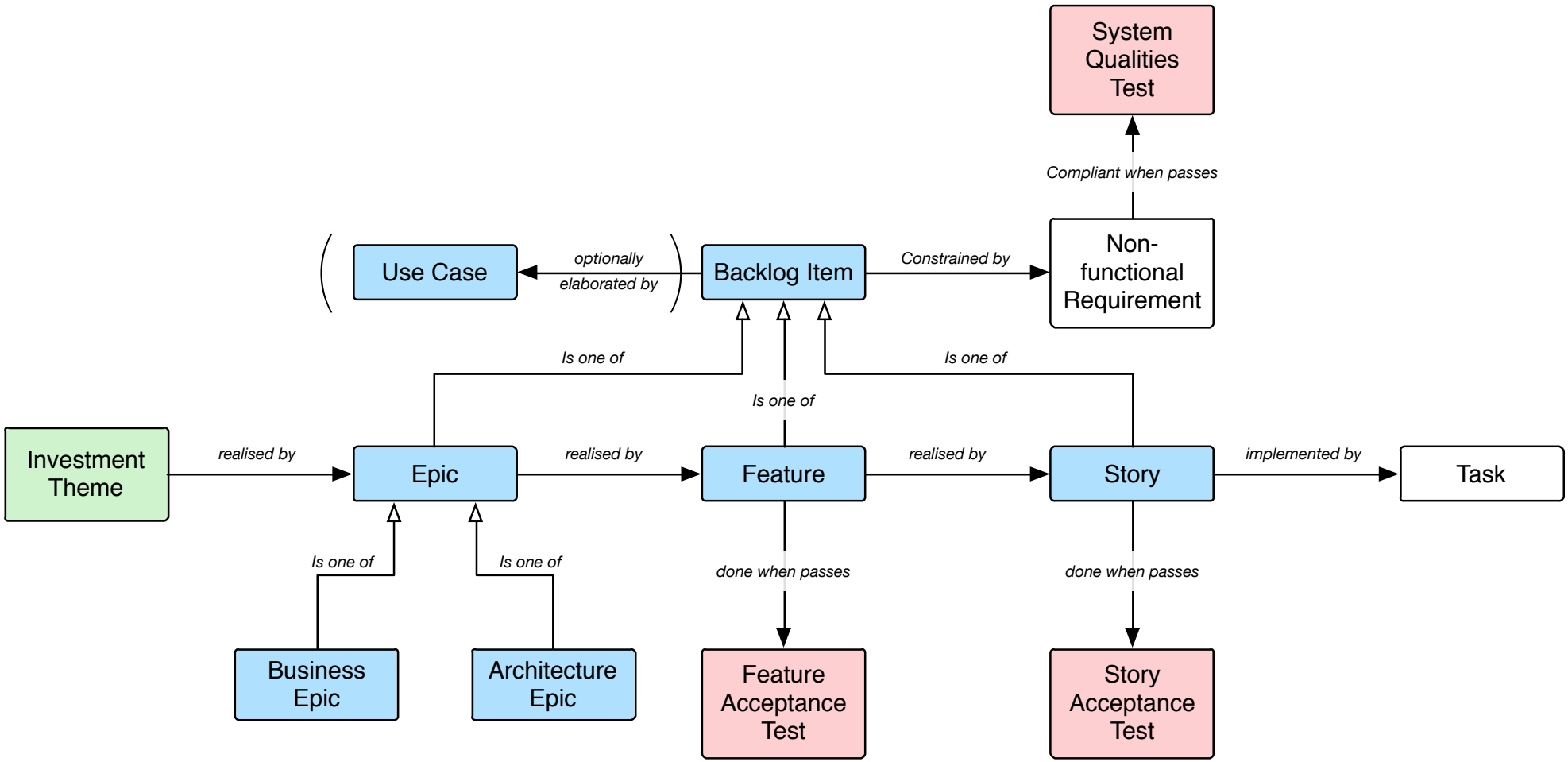
Estimation


?
0
1/2
1
2
3
5
8
13
20
40
100
∞
Books by Mike Cohn
Our Agile Courses
Succeeding with Agile

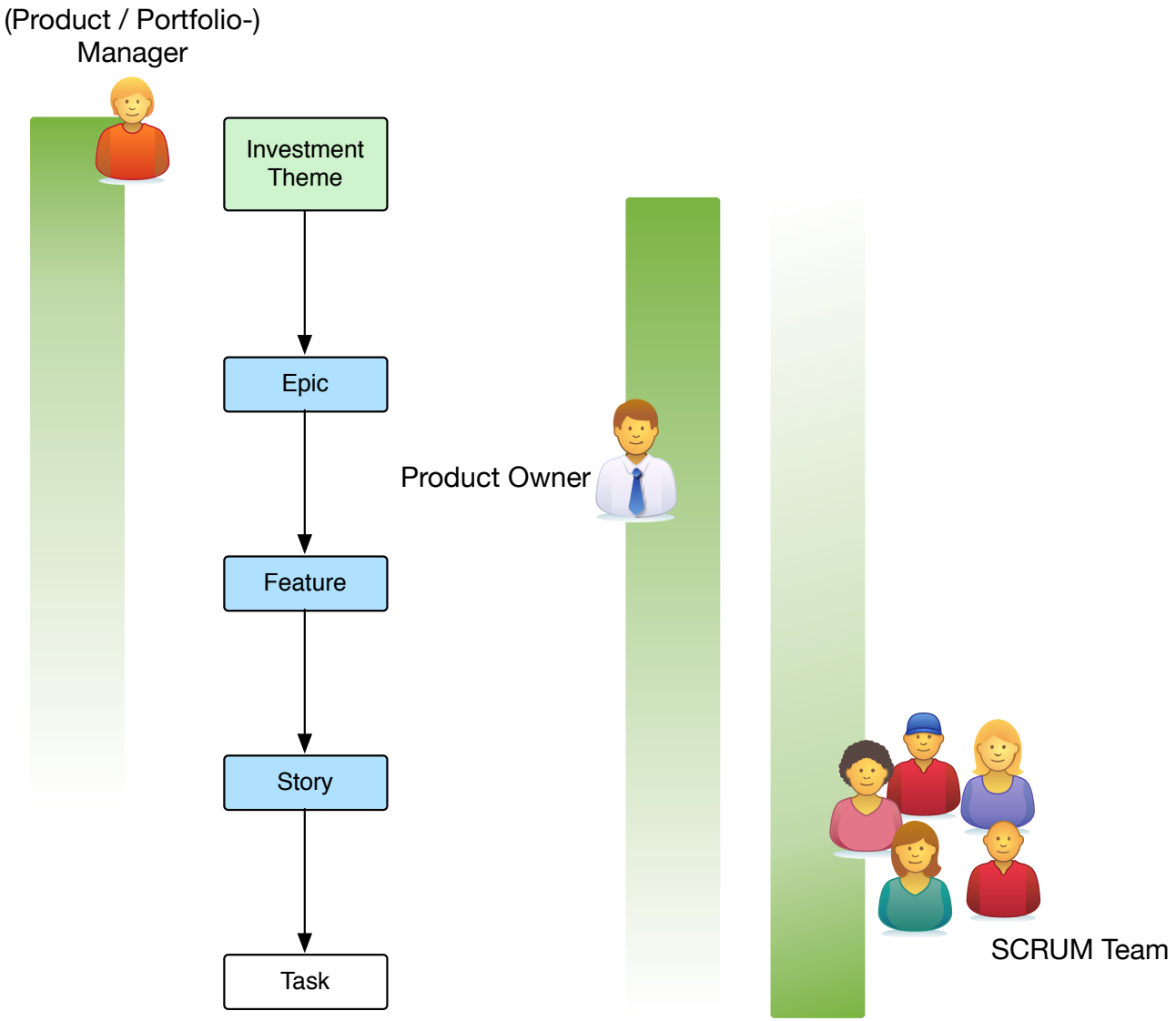
SAFE Framework

- Vision and Roadmap
- Release Management
- Deployment
- Resource Management
- Cross-team tasks

Full Enterprise Requirement Model (SAFE Framework)



Roles and “Stories”

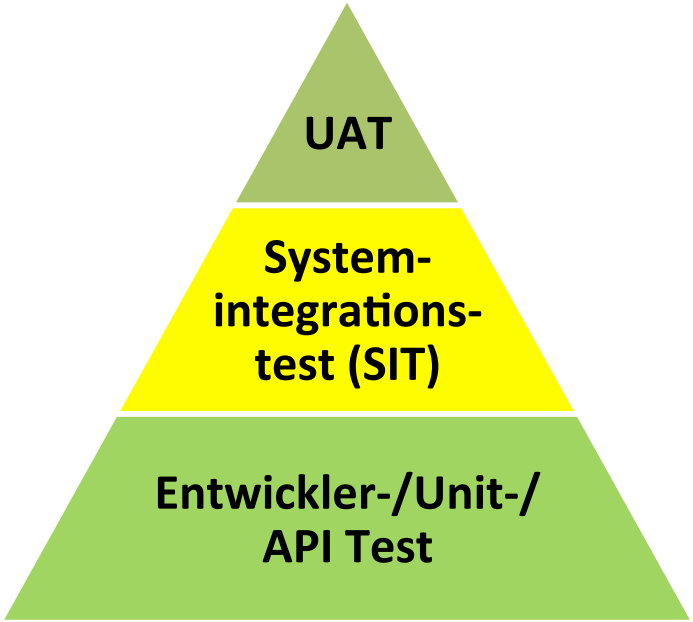


Teststufen und -arten

- **Entwickler-/ Unit-/ API Test**
  - Prüfung des Testobjekts (Source Code, programmbezogene Objekte und Methoden, etc.) auf Erfüllung der Vorgaben der geforderten Funktionalität
  - Anwendung von White-Box-Testverfahren
  - Frühzeitige Anwendung von Progressionen sowie Regressionen

- **Systemintegrationstest**
  - Überprüfung des Zusammenspiels der Einzelkomponenten
  - Aufdeckung von Abweichungen und Fehlerzuständen zwischen integrierten Systemteilen
  - Verifikation der progressiven sowie regressiven fachlich orientierten Prozessabläufe

- **User Acceptance Test**
  - Überprüfung der IT Lieferungen auf Übereinstimmung der vereinbarten Vorgaben
  - Einnahme der Rolle Tester durch den Fachbereich
  - Einsatz von Black-Box-Testverfahren, welche progressiv als auch regressiv ausgeprägt sind
  - Zustandsverifikation zur Abnahmeentscheidung



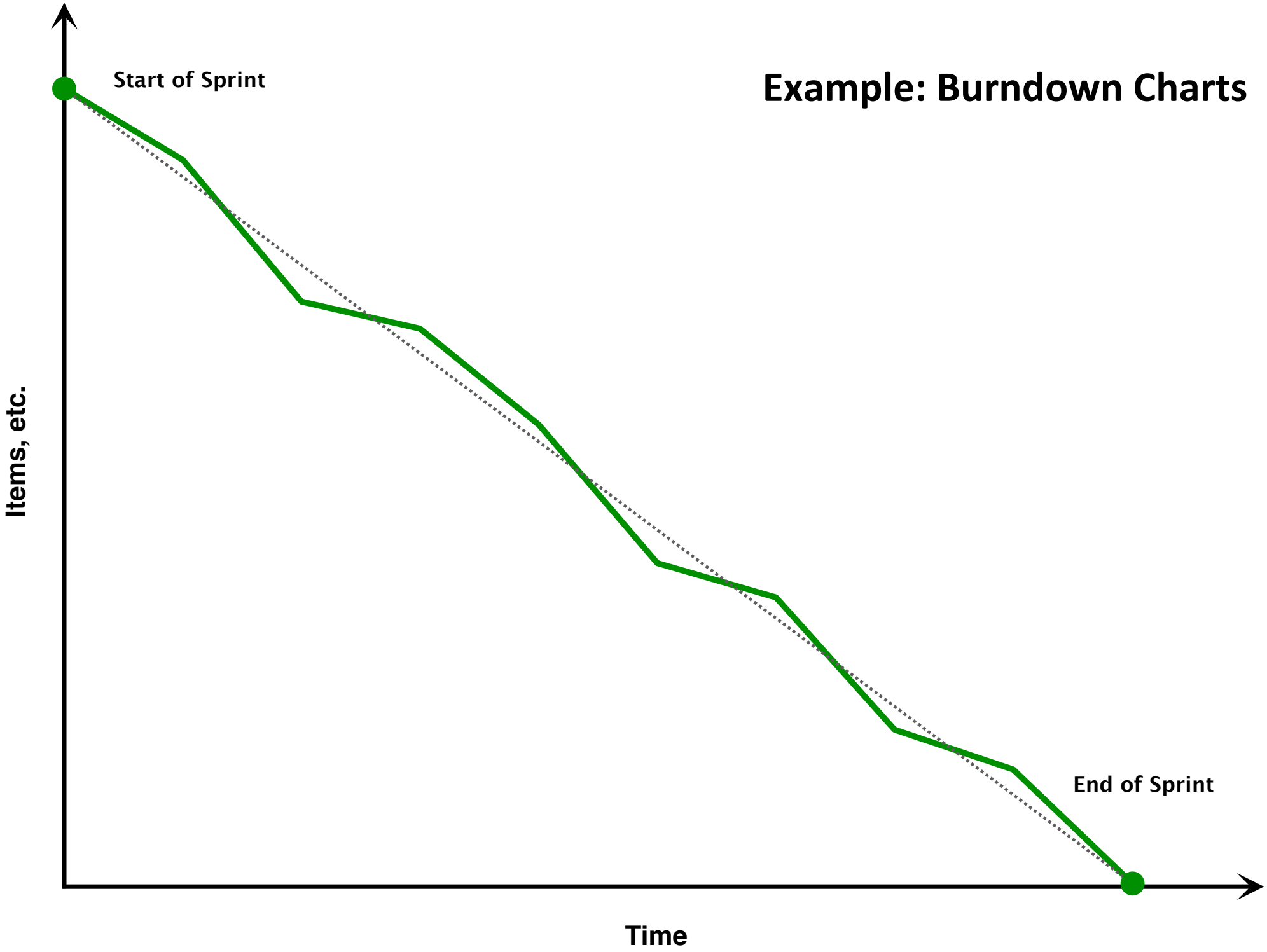
- **Produktionsabsicherung / Betriebliche Maßnahmen**
  - Absicherung des Produktiven Betriebes durch lesende bzw. schreibende Smoketests
  - Überprüfung der Zustandsänderung welche durch eine Release bzw. Lieferung verursacht wurde

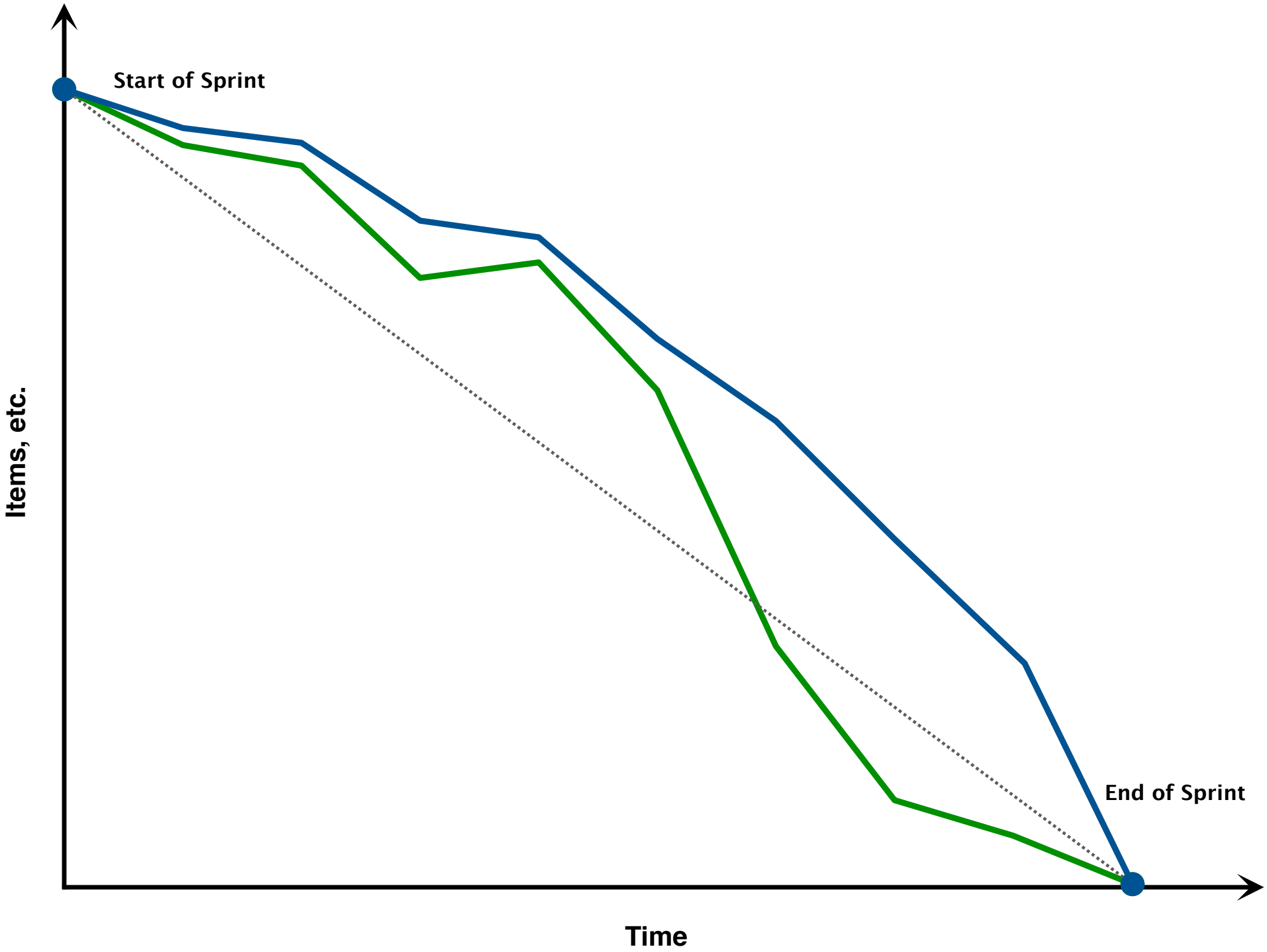
Transparency

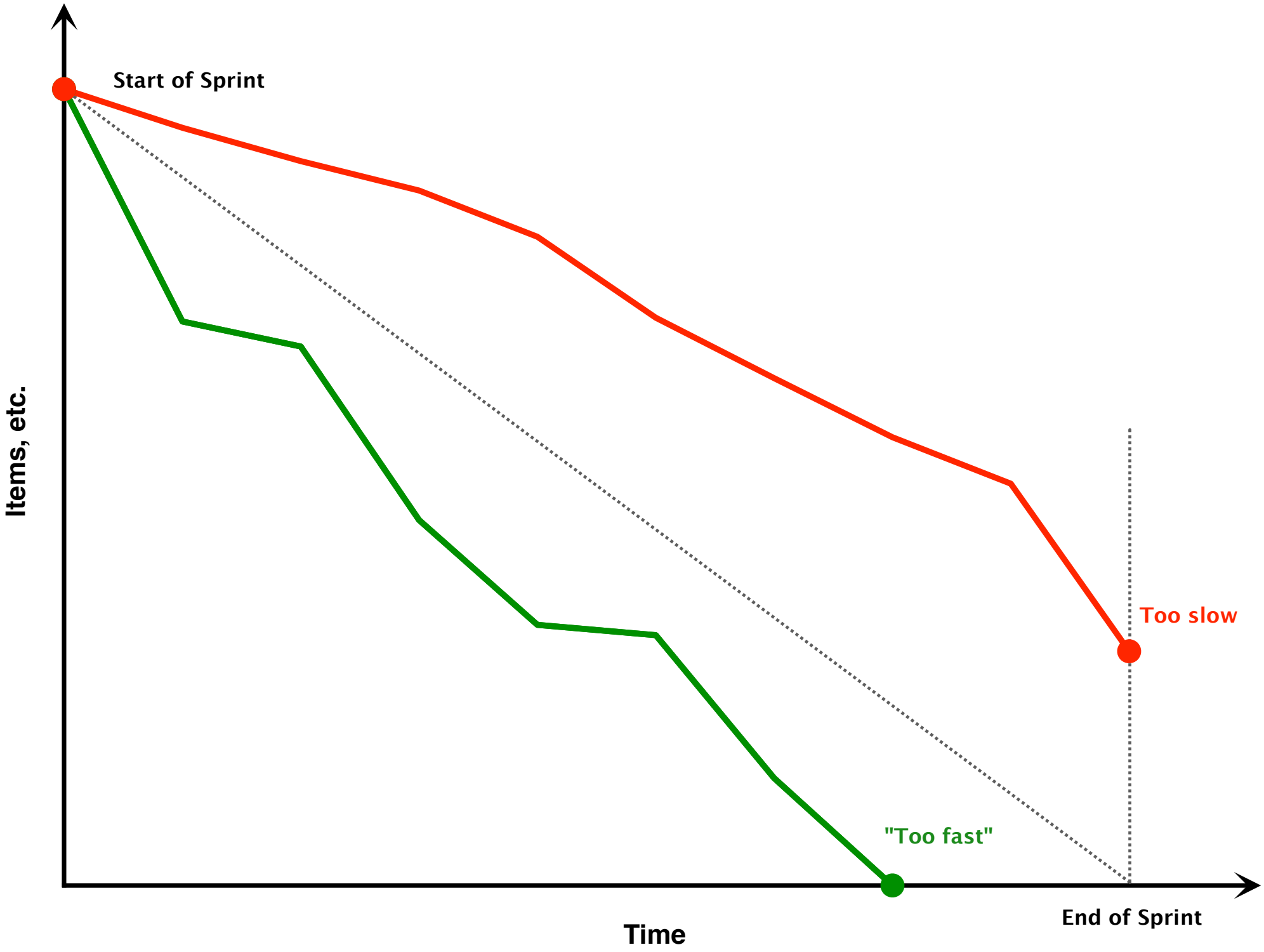
Controlling: Burn Down Charts

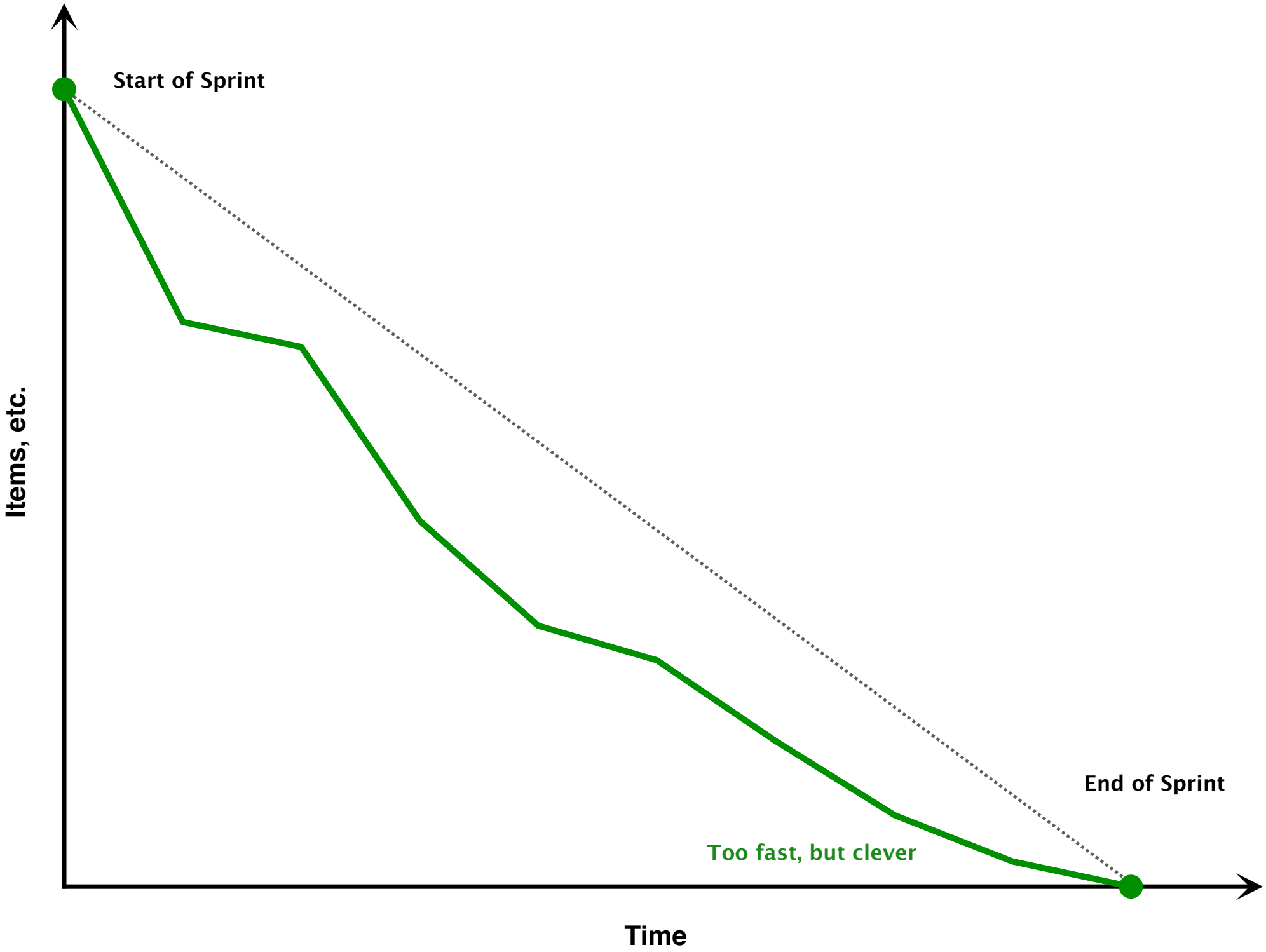
- One option to visualise progress
- Y-axis: items/features (e.g. “Story Points”, person days)
- X-axis: time
- Update after each sprint
- Improves team self-assessment
- Realistic estimation of project duration and assessment of changes in project scope
- Velocity (KPI): work per iteration: measured, not estimated (!)

Example: Burndown Charts

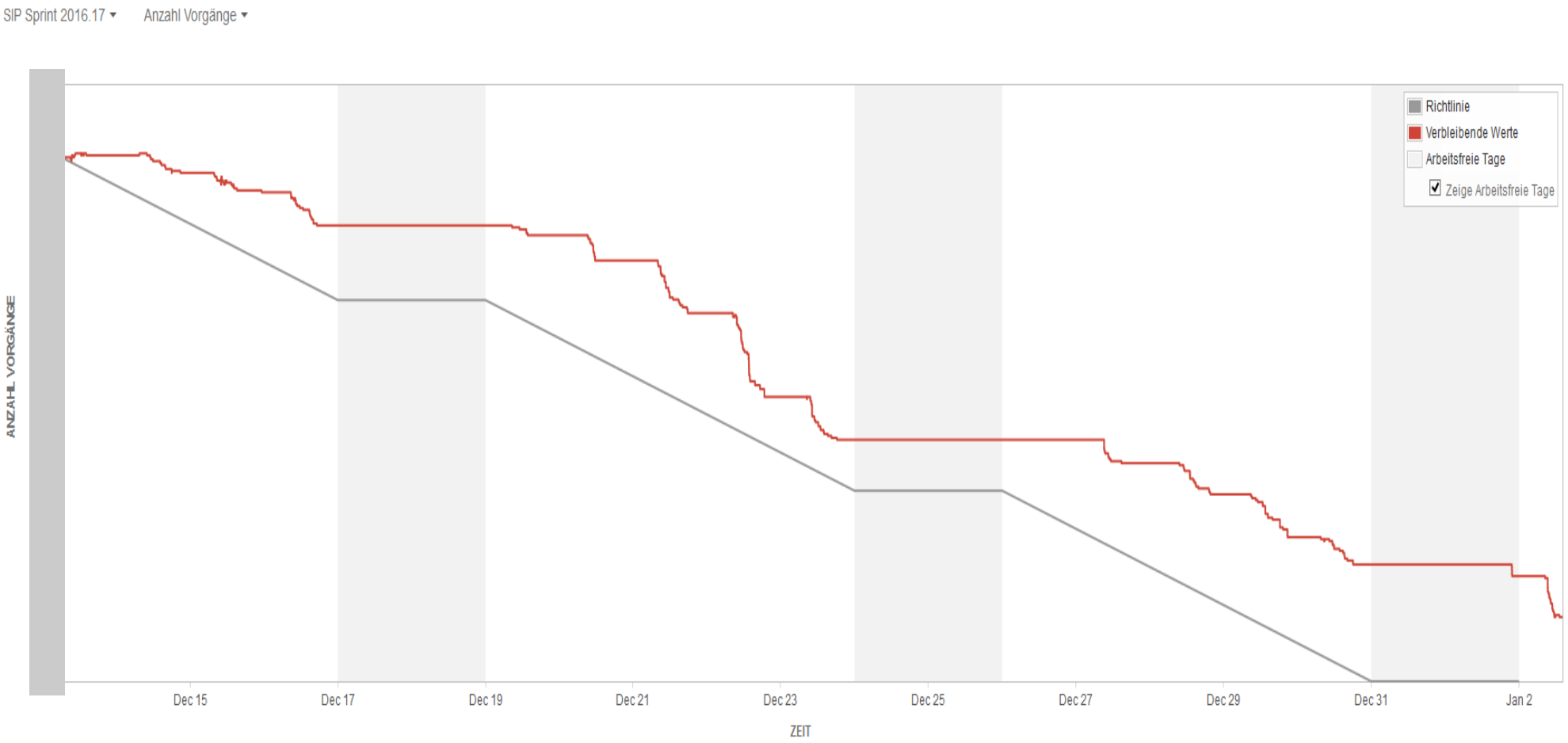



Start of Sprint
End of Sprint
Items, etc.
Time


Start of Sprint
Items, etc.
Too slow
"Too fast"
Time
End of Sprint


Start of Sprint
Items, etc.
End of Sprint
Too fast, but clever
Time

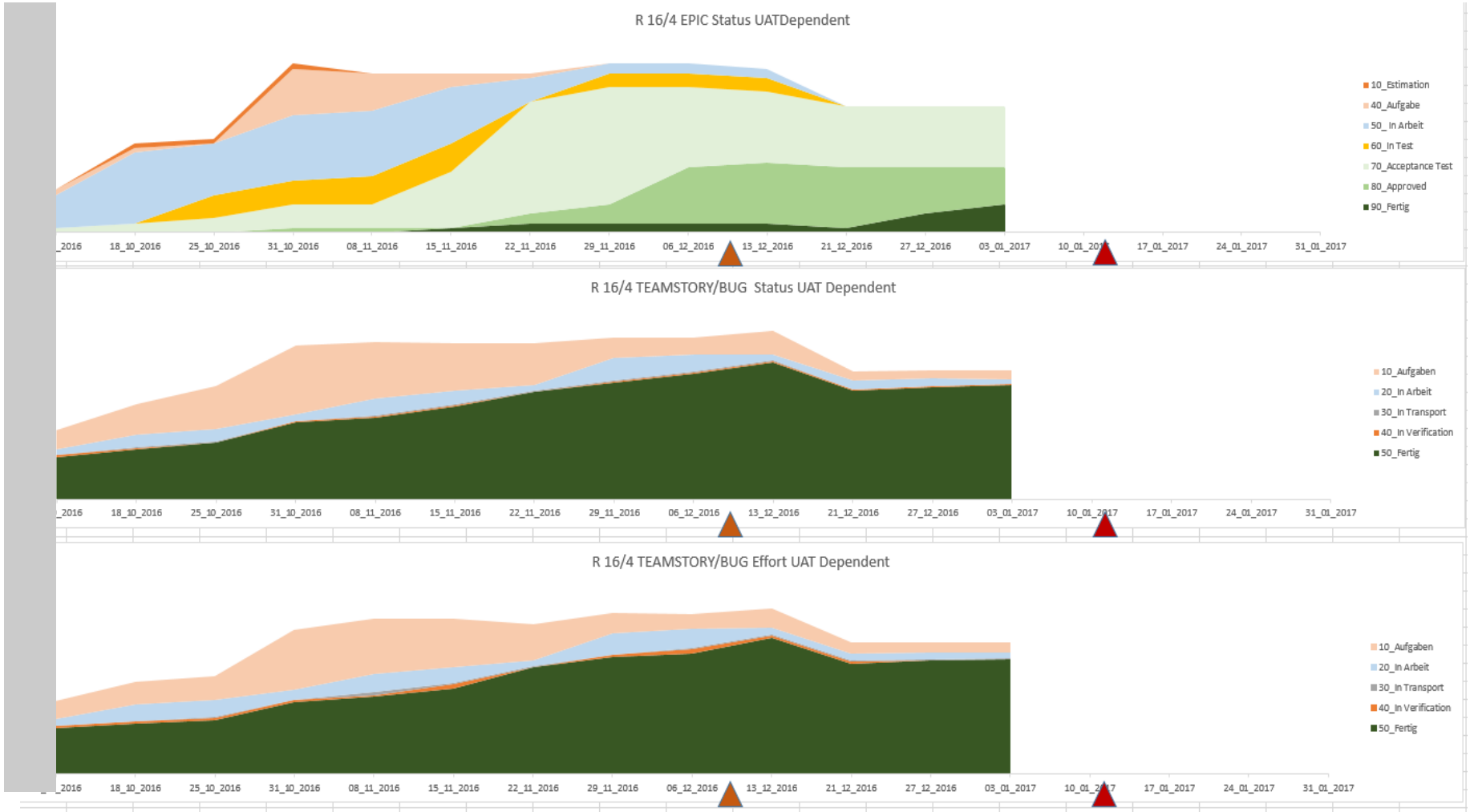
Example Burn-Down Chart



Example: Sprintrate Teams

ALL TASKS										
AS	gesamt	offen	fertig	obsolet	moved	offen	fertig	obsolet	moved	Sprintrate %
						0%	92%	8%	0%	100%
						5%	70%	15%	10%	82%
						0%	88%	0%	13%	88%
						7%	79%	14%	0%	92%
						17%	81%	2%	0%	83%
						0%	76%	22%	2%	97%
						0%	100%	0%	0%	100%
						0%	100%	0%	0%	100%
						0%	83%	0%	17%	83%
						0%	71%	0%	29%	71%
						13%	88%	0%	0%	88%
						14%	86%	0%	0%	86%
						0%	100%	0%	0%	100%
						20%	80%	0%	0%	80%
						41%	39%	10%	10%	43%
						13%	75%	13%	0%	86%
						43%	57%	0%	0%	57%
						40%	0%	60%	0%	0%
						6%	94%	0%	0%	94%
						0%	100%	0%	0%	100%
						100%	0%	0%	0%	0%
						100%	0%	0%	0%	0%
SUMME	331	42	247	26	16	13%	75%	8%	5%	81%

Example: Release CFD (UATdependent)



Progress / Costs / Budget

- Reporting of progress often quite difficult
- Progress according to defined scope is (comparatively) easy but
- Is development in budget?
  - Time recording
  - Different cost per day per employee
  - Internal / external team members
  - Other costs (development server, licenses, etc.)
  - Organisation of IT: cost centre, commercial entity, etc.
- Actuals versus planning (do I know *who* is going to do *what* taks? *Quite the opposite of agile*)
- Soon, administration tasks for (dev teams) becomes all but *lean* and *self-organised*

Conclusion
Agile Metrics (KPIs)

- Different options to visualise progress
  - Burn-Down Charts
  - Prozess-flow visualisation
  - Cumulative Flow Diagrams
- Various KPIs
  - Velocity (aggregated): *items per iteration*
  - Velocity per work type
  - Cycle Time/Lead Time: *Average completion time of one item*
  - Identification of bottlenecks (queue length)
  - Defect Rates
- Usefulness of metrics/KPIs depending on
  - Project environment
  - Concrete process implementation
  - Maturity of teams / process implementation
- Aggregate Metrics
  - Progress metrics over team boundaries
  - Progress metrics including actuals vs. budget?
- Cross team metrics / KPIs, e.g. story points?
  - Can be used but has risks

ADDITIONAL THOUGHTS

Challenges and Risks

- Introduction / **change management: CEO / board support essential**
- Process not implemented properly / consequently
- Role of product owner / customer
- Team structure not clear enough
- Team roles still focused on “silos”
- Lack of **trust**
- Lack of transparency or different understanding of transparency
- KPIs and reporting that is not benefiting teams and management alike: *introduction of KPIs often backfires as it steers people to fulfil the KPI not the actual intention behind it*
- **Cost/backlash of transparency**: transparency is the evil brother of self-organisation and trust
- Complexity of architecture and systems

How does management (and most of us) see IT?

Confidence Building

- Change management as part of the process
- Get external experts when needed and get forcefully rid of legacy (this is the hardest part)
- Frequent releases and demos
- Customer relationship
- Backlog transparency
- Testing and QA
- Agile metrics / KPIs (that helps teams and management alike – **very** difficult to establish)
- *But from the other side: confidence and trust can be seen as the opposite of transparency*
- Audits

Messen

»Die messbare Seite der Welt ist nicht die Welt, *es ist die messbare Seite der Welt.*«

Martin Seel

Vs. »Messen was messbar ist, messbar machen was nicht messbar ist« — Effizienzsteigerung

(Galileo, ... oder Archimedes?)

Creativity?

»you don’t get creativity for free. You need people to be able to sit back, put their feet up, and think.«

»An organisation that can accelerate but not change direction is like a car that can speed up but not steer. In the short run, it makes lots of progress in whatever direction it happened to be going. In the long run, it’s just another road wreck.«

Tom De Marco

Conclusion: Cornerstones

- Agile or plan-driven?
- Agile is the opposite of sloppy!
- Transparency over following a (long term, improbable) plan
- Agile – not only for small teams
- Requirements and customers
- Roles and processes
- Balance Measurment and Self-Organisation

Agile Softwareentwicklung im Konzernumfeld

Dr. Alexander Schatten
alexander@schatten.info
http://www.schatten.info