

Kapitel 13.

Agile Softwareentwicklung am Beispiel Extreme Programming (XP)

Stand: 22.01.2020

Folie 13: Eingebildet
Folie 27: Ergänzt

Was sind „Agile Methodologien“?

- Eine **Methodologie** ist eine bestimmte Art und Weise den Softwareentwicklungsprozess zu organisieren. Sie legt fest:
 - ◆ Was wir tun
 - ◆ Wann wir es tun
 - ◆ Welche Werkzeuge wir benutzen
 - ◆ Wie wir den Prozess planen
 - ◆ Wie wir den Prozess kontrollieren
 - ◆ ...
- Eine **Agile Methodologie** ist eine Methodologie die versucht die Entwicklung zu beschleunigen durch:
 - ◆ **Zusammenarbeit mit dem Kunden** ↔ Anstelle von Vertragsverhandlungen
 - ◆ **Auf Änderungen reagieren** ↔ Anstatt einem Plan zu folgen
 - ◆ **Funktionierende Software** ↔ Anstelle von umfangreichen Dokumenten
- Beispiel: „**Extreme Programming**“

Was ist "eXtreme" Programming?

"Extreme Programming setzt **bekannte Prinzipien** und Praktiken extrem konsequent um."

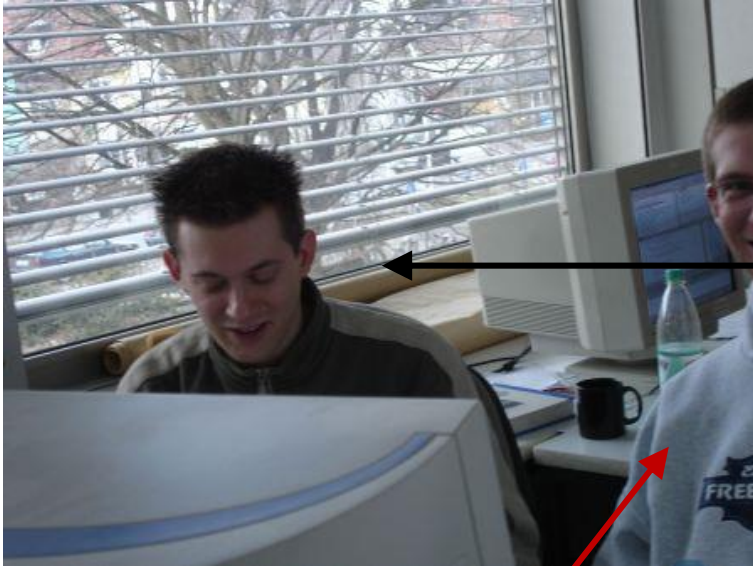
Bekannte Prinzipien und Praktiken

- Feedback ist gut
- Code reviews sind gut

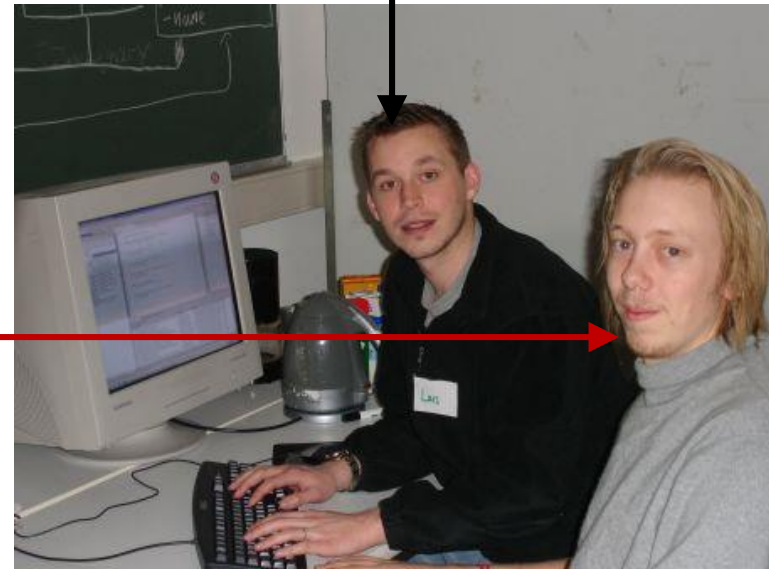
... extrem konsequent umgesetzt

- Kunde ist ein Teil des Teams
 - ◆ permanentes Kundenfeedback
- Pair Programming
 - ◆ permanente Code Reviews!

Pair Programming: 2 Partner, 1 Computer



1 Partner programmiert.



Der 2. Partner überprüft den Code.
... oder plant voraus.



Pair Programming

- Szenario
 - ◆ ein Rechner, eine Tastatur, eine Maus
 - ◆ zwei Programmierer
 - ◆ abwechselnde Rollen-Teilung
- Implementierer-Rolle
 - ◆ implementiert
- Code-Reviewer-Rolle
 - ◆ Kann das so funktionieren?
 - ◆ Gibt es Testfälle, die wir noch nicht bedacht haben?
 - ◆ Kann man das Problem durch eine Vereinfachung des Designs lösen?
 - ◆ ...
- Rollen jederzeit änderbar (vor allem, wenn „Implementierer“ müde wird)
- Paare jederzeit änderbar (tyischerweise nach Erledigung einer Task)

Was ist "eXtreme" Programming?

"Extreme Programming setzt bekannte Prinzipien und Praktiken extrem konsequent um."

Bekannte Prinzipien und Praktiken

- Feedback ist gut
- Code reviews sind gut
- Testen ist gut
- Integrationstests sind gut

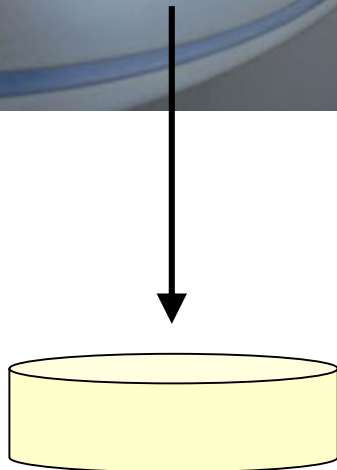
... extrem konsequent umgesetzt

- Kunde ist ein Teil des Teams
 - ◆ permanentes Kundenfeedback
- Pair Programming
 - ◆ permanente Code Reviews!
- Permanentes testen!
 - ◆ Programmierer: Unit tests
 - ◆ Kunde: Funktionstests
- Kontinuierliche Integration
 - ◆ So oft wie möglich

Continuous Integration: Integriere so oft wie möglich!



Integriere nur wenn
alle
Tests erfolgreich waren!



Gemeinsames Repository



Was ist "eXtreme" Programming?

"Extreme Programming setzt bekannte Prinzipien und Praktiken extrem konsequent um."

Bekannte Prinzipien und Praktiken

- Feedback ist gut
- Code reviews sind gut
- Testen ist gut
- Integrationstests sind gut
- Design ist wichtig
- Einfachheit ist gut
- Kurze Iterationen sind gut

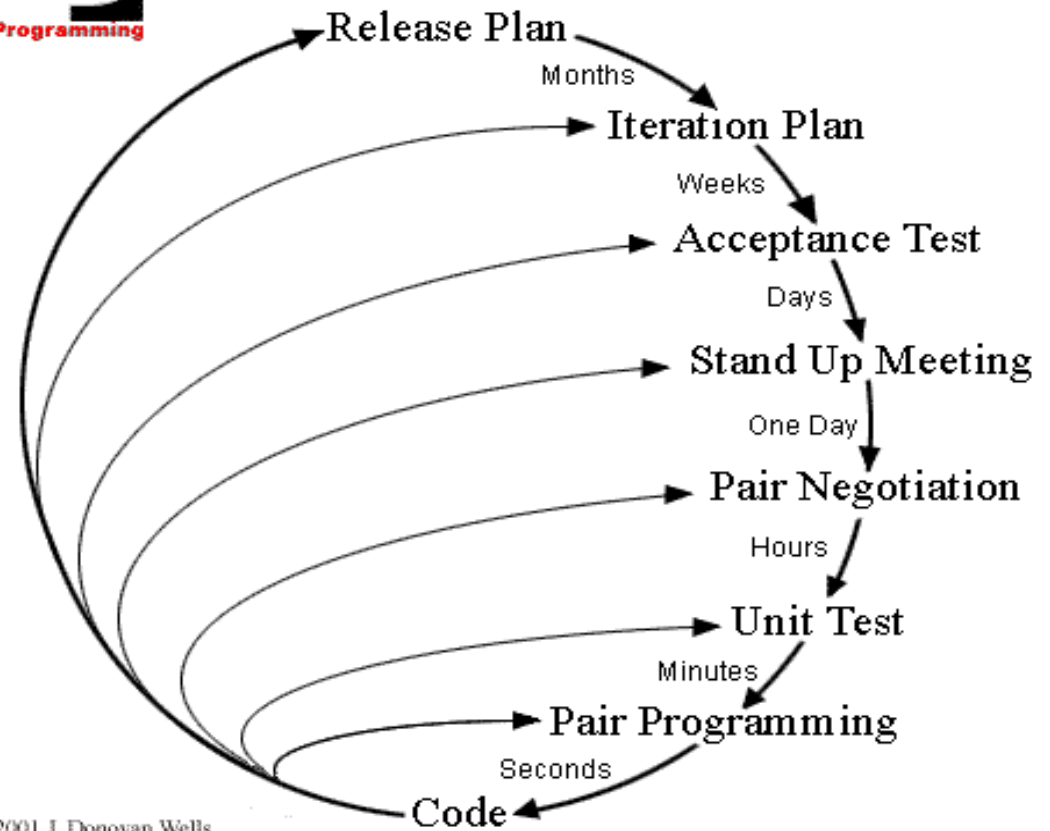
... extrem konsequent umgesetzt

- Kunde ist ein Teil des Teams
 - ◆ permanentes Kundenfeedback
- Pair Programming
 - ◆ permanente Code Reviews!
- Permanentes testen!
 - ◆ Programmierer: Unit tests
 - ◆ Kunde: Funktionstests
- Kontinuierliche Integration
 - ◆ So oft wie möglich
- Refactoring
 - ◆ permanentes (Re)Design
- ziehe einfache Lösungen vor
 - ◆ Refaktoriere später, wenn nötig
- „Planning Game“
 - ◆ Sehr kurze Iterationen

Agile Praktiken



Planning/Feedback Loops



Story Cards

USERS SHOULD BE ABLE TO UPLOAD MULTIPLE PHOTOS AT ONCE

- Test with JPEG, PNG, GIF (supported)
- Test with an unsupported image format
- Test with Flash not present
- Test with more than 20 MB of total POST data

StoryTag: DocBookToHTML Release: Book Priority: 1

Author: Joanne on: 2/21/02 Accepted: 3/17/02

Description: Make the DocBook files readable and printable.

Considerations: HTML has some drawbacks: Estimate: 4.1

- Printed version is not production quality.
- Footnotes can't appear at end of page.

Who	Task	Est.	Done
Rob	Simple tags: <chapter>, <title>, <para>	1	2/24
Rob	Asymmetrical tags: <attribution>	1	3/3
Rob	Contextually related tags: <title>	1	3/11
Rob	Stateful output: <footnote>	1	3/14
Joanne	Acceptance Test: Print the first chapter	0.1	3/17

Story Cards



Planning Game: Schätze Kosten und Risiken → Plane ein Release

Schritte des „Planning Game“

- Identifiziere „Stories“ (= Funktionalität die der Kunde benötigt)
- Bestimme eine Priorität für jede Story

Kunde

- Schätze die Kosten für jede Story
- Schätze die Risiken für jede Story

Programmierer

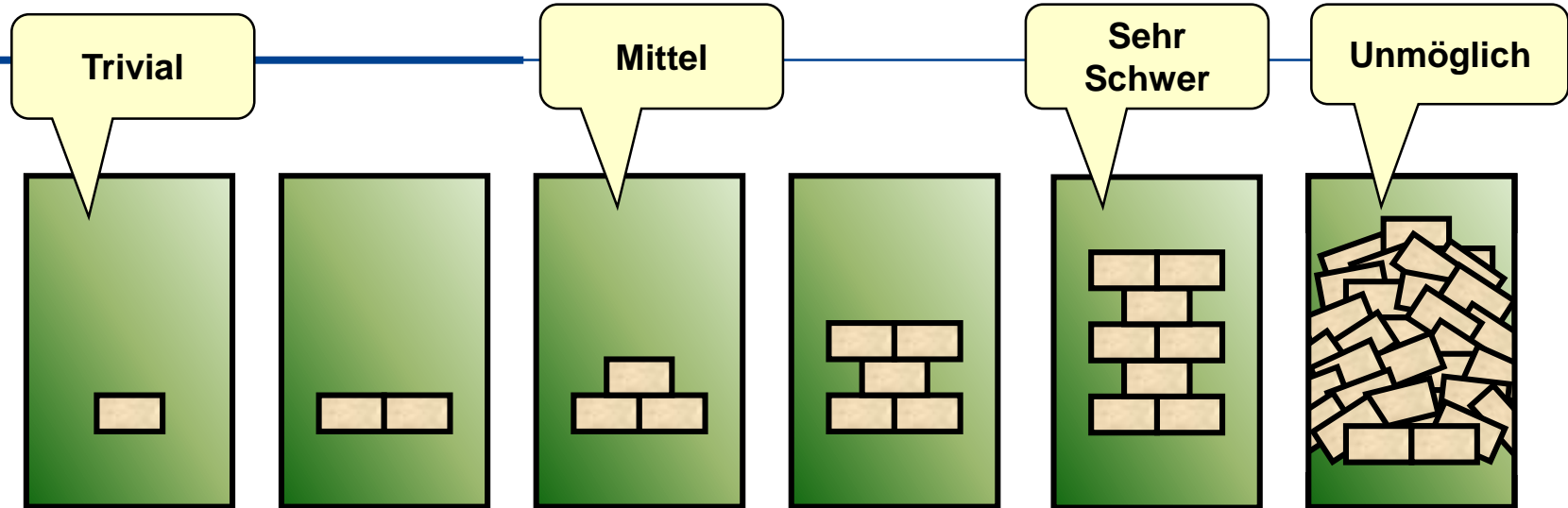
- Lege die zu implementierende Funktionalität fest
 - ◆ 1 Iteration = 5 Arbeitstage
 - ◆ Wähle die Stories aus, die in der nächsten Iteration realisierbar sind
 - ◆ Stelle die anderen zurück

Gemeinsam

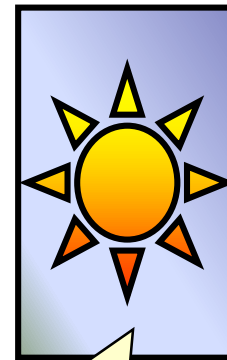
Planspiel

- SW-Entwicklung als Dialog zwischen
 - ◆ Wünschenswertem (Kunden-Sicht)
 - ◆ Machbarem (Programmierer-Sicht)
- Kunden entscheiden über
 - ◆ Funktionsumfang
 - ◆ Prioritäten
 - ◆ Zusammensetzung von Releases die einen echten Mehrwert bietet
 - ◆ Zeitpunkt von Releases
- Programmierer entscheiden über
 - ◆ Aufwandsschätzungen
 - ◆ Technische Folgen eines Kundenwunsches (z.B. DB-Auswahl)
 - ◆ Prozess (interne und externe Team-Zusammenarbeit)
 - ◆ Interne Zeitpläne
 - ⇒ was passiert innerhalb eines Release zuerst?
 - ⇒ risiko-behaftete Aspekte vorziehen soweit es die Issue-Abhängigkeiten erlauben

Planning Game: Schätze Kosten und Risiken → Plane ein Release



Risiko, daß der benötigte Aufwand viel **höher** sein könnte.



Chance, daß der wirkliche Aufwand viel **niedriger** sein könnte.

Das Planning Game (basierend auf <http://www.xp.be/xpgame.html>)

● Ablauf

- ◆ Kunde liest eine Story vor
- ◆ Entwickler stellen letzte Fragen
- ◆ Jeder Entwickler wählt **verdeckt** die seinen Schätzungen entsprechenden Karten
- ◆ Wenn alle fertig sind, werden die Karten gleichzeitig umgedreht.
- ◆ Übereinstimmung:
Mit nächster Story fortfahren
- ◆ Falls nicht:
Diskussion derer, die in ihrer Schätzung am weitesten auseinander liegen

● Nutzen

- ◆ Beteiligt das ganze Team
- ◆ Beschleunigt die Schätzung der Stories → Diskussion nur wenn nötig!!!



Issue Tracking System „Jira“: Übersicht

http://roots.iai.uni-bonn.de/jira32/secure/IssueNavigator.jspa?mode=hide&requestId=10090



Research on Object-Oriented Technologies and Systems

User: Daniel Speicher [History](#) | [Filters](#) | [Profile](#) | [Log Out](#)

[HOME](#) [BROWSE PROJECT](#) [FIND ISSUES](#) [CREATE NEW ISSUE](#) [ADMINISTRATION](#)

QUICK SEARCH:

Filter: [View](#) [Edit](#) [New](#) [Manage](#)

Name:

Prompter stories

Description:

All stories of the project prompter.

Summary

- Project:** [Prompter](#)
- Issue Type:** Story
- Sorted by:** Fix Version/s ascending, then Summary ascending, then Priority ascending

Operations

- [Rename](#) current filter
- [Save as](#) new filter
- [Subscriptions](#) for current filter
- [Share](#) current filter

Issue Navigator -- Prompter stories

Displaying issues **1** to **20** of 22 matching issues.

Current View:

[Browser](#) | [Printable](#) | [XML](#) | [Full Content](#) | [Excel \(All fields\)](#) | [Current fields](#)

- [Bulk Change](#)
- [Configure](#) yo

1 | 2 | [Next >>](#)

T	Key	Summary	Assignee	Pr	Status	Res	Created	Updated	Fix Version/s	Orig
	PR-3	S00 Prepare development infrastructure	Daniel Speicher	↑	Closed	Fixed	02/Nov/05	06/Dec/05	0.2M1	
	PR-1	S01 Easy Checkout	Daniel Speicher	↑	Closed	Fixed	02/Nov/05	15/Jan/06	0.2M1	16 h
	PR-14	S02 SPIKE Interaction categorisation + indexing	Unassigned	↑	Closed	Fixed	02/Nov/05	05/Dec/05	0.2M1	
	PR-17	S03 Prototype executably locally on laptop	Unassigned	↑	Open	UNRESOLVED	02/Nov/05	07/Nov/05	0.2M1	
	PR-11	S04 Anwendungen ohne Kategorisierung	Unassigned	↑	Open	UNRESOLVED	02/Nov/05	06/Feb/06	0.2M1	
	PR-13	S05 Static test coverage >= 30%	Unassigned	↑	Open	UNRESOLVED	02/Nov/05	18/Nov/05	0.2M1	
	PR-53	S14 Move scenario specific data into separate projects	Unassigned	↑	Open	UNRESOLVED	09/Jan/06	13/Feb/06	0.2M1	12 h
	PR-55	S15 Create Gui for CreateData	Bernd Wahlen	↑	Closed	Fixed	09/Jan/06	10/Jan/06	0.2M1	3 h
	PR-57	S16 Gui for manual Locaton-Change	Bernd Wahlen	↑	Closed	Fixed	09/Jan/06	13/Jan/06	0.2M1	16 h
	PR-71	S18 Remove static dependency on the indexing bundle from the mail and notes client	Unassigned	↑	Open	UNRESOLVED	06/Feb/06	14/Feb/06	0.2M1	

Issue Tracking System „Jira“: Story



Research on Object-Oriented Technologies and Systems

User: Daniel Speicher [History](#) | [Filters](#) | [Profile](#) | [Log Out](#)

[HOME](#) [BROWSE PROJECT](#) [FIND ISSUES](#) [CREATE NEW ISSUE](#) [ADMINISTRATION](#)

QUICK SEARCH:

Issue Details [\[XML\]](#)

Key: [PR-1](#)
Type: Story
Status: Closed
Resolution: Fixed
Priority: Major
Assignee: [Daniel Speicher](#)
Reporter: [Daniel Speicher](#)
Votes: 0 [\(View\)](#)
Watchers: 0 [\(View\)](#)

Available Workflow Actions

[Reopen Issue](#)

Operations

[Assign](#) this issue
 [Clone](#) this issue
 [Comment](#) on this issue
 [Delete](#) this issue
 [Link](#) this issue to another issue
 [Move](#) this issue
 [Voting:](#)
You cannot vote or change your vote on resolved issues.

[Watching:](#)
You are not watching this issue.
[Watch it](#) to be notified of changes

[Worklog:](#)
Worked on this issue? [Log work done](#)

Prompter
 S01 Easy Checkout
Created: 02/Nov/05 02:26 PM Updated: 15/Jan/06 02:09 PM

Component/s: None
Affects Version/s: None
Fix Version/s: [0.2M1](#)

[Return to search](#)
"Prompter stories"
Issue 2 of 22 issue(s)
[<< Previous](#) | [PR-1](#) | [Next >>](#)

Original Estimate:	2 days	Remaining Estimate:	2 days	Time Spent:	Unknown
---------------------------	--------	----------------------------	--------	--------------------	---------

Issue Links:
[Manage Links](#)

Story/Task Relationship

This issue contains:

- [PR-2](#) S01-T01 Extract hard coded configurat...
- [PR-40](#) S01-T02 Konfiguration zentralisieren
- [PR-42](#) S01-T05 Dokumentation
- [PR-45](#) S01-T03 provide SWT-Bundles
- [PR-46](#) S01-T04 OSGi Knopf. Run Configuration
- [PR-24](#) S01-T06 Rename BundleSWT to BundleSWT...
- [PR-26](#) S01-T07 Move native-code libraries fr...
- [PR-34](#) S01-T08 SPIKE: automatic MAC address ...
- [PR-36](#) S01-T09 Create empty files if missing
- [PR-38](#) S01-T10 Discuss name and responsibili...
- [PR-39](#) S01-T11 Write a short tutorial how to...
- [PR-30](#) S01-T12 Check the documentation for u...




Description

It shall be as simple as possible to checkout the sources of the project and run the bundles. Everything a developer needs to know for configuration is clearly documented in our wiki (<http://roots.iai.uni-bonn.de/prompter/>). Dependencies on the J9 and eSWT shall be eliminated unless they need a global restructuring. All configuration informations that are still hard coded at the moment shall be extracted to a configuration file.

Issue Tracking System „Jira“: Task

[HOME](#) [BROWSE PROJECT](#) [FIND ISSUES](#) [CREATE NEW ISSUE](#) [ADMINISTRATION](#)

Issue Details [\[XML\]](#)

Key: [PR-2](#)
Type:  Task
Status:  Closed
Resolution: Fixed
Priority:  Major
Assignee: [Kai-Lin Pang](#)
Reporter: [Daniel Speicher](#)
Votes: 0 [\(View\)](#)
Watchers: 0 [\(View\)](#)


Available Workflow Actions

- [Reopen Issue](#)

Operations

- [Assign](#) this issue [\(to me\)](#)
- [Clone](#) this issue
- [Comment](#) on this issue
- [Delete](#) this issue
- [Link](#) this issue to another issue
- [Move](#) this issue

Prompter

 **S01-T01 Extract hard coded configuration information**
Created: 02/Nov/05 02:31 PM Updated: 05/Nov/05 05:03 PM

Component/s:	None
Affects Version/s:	None
Fix Version/s:	0.2M1




Original Estimate:	Unknown	Remaining Estimate:	Unknown
---------------------------	---------	----------------------------	---------

Issue Links:

[Manage Links](#)

Story/Task Relationship

This issue *is part of*.

 [PR-4](#) S01 Easy Checkout  

Description

Create a class in the BundleCSICommon to access a properties file (or xml configuration file). Replace information to an access to this class.

[All](#) [Comments](#) [Work Log](#) [Change History](#)

Comment by [Kai-Lin Pang](#) [05/Nov/05 05:03 PM] [Delete](#)

All important hard coded references (ONLY Bundles) are replaced by a properties file called "config.ini"

Werkzeuge: Story Tracking als Beispiel

- Papier
 - ◆ (+) Besserer Überblick über den Gesamtzustand
 - ◆ (+) Leichter zu modifizieren.
 - ◆ (+) Ein Bild sagt mehr als tausend Worte.
 - ◆ (-) Verschwinden nach dem Kurs.
 - ◆ (-) Keine Anfragen möglich.

- Jira (Web-basiertes Issue Tracking)
 - ◆ (+) Persistente Repräsentation der Entwicklungsgeschichte
 - ◆ (+) Lange Zeit sichtbar
 - ◆ (-) Überblick nicht so klar
 - ◆ (-) Eintragen der Tasks / Stories aufwändiger, keine Zeichnungen

Eine genauere Betrachtung

Einflussfaktoren auf den SW-Entwicklungsprozess

- Kosten
- Qualität
- Zeit
- Funktions-Umfang

Grund-Zusammenhang

Durch Festlegung von drei beliebigen Faktoren ist der Vierte mit festgelegt !

Konsequenz

Der Kunde kann höchstens drei Faktoren nach seinem Wunsch bestimmen.
Die Entwickler müssen ihm den Einfluss auf den vierten Faktor erläutern!

Einfluss des Kostenrahmens

- zu wenig Geld
 - ◆ keine effektive Entwicklung
- mehr Geld
 - ◆ bessere Ausstattung, Umgebung, Ausbildung, ...
- aber Geld allein macht kein erfolgreiches Projekt
 - ◆ "40 Programmierer"-Anekdote
- besser
 - ◆ Projektgröße schrittweise anpassen
- Problem
 - ◆ Status- / Prestige-Denken von Projektleitern
 - ⇒ "Ich hab ein 150-Personen-Projekt..."

Einfluss des Zeitrahmens

- zu wenig Zeit
 - ◆ schlechte Qualität
 - ◆ geringer Umfang
 - ◆ hohe Kosten
- mehr Zeit
 - ◆ bessere Qualität
 - ◆ mehr Funktionalität
- aber zu viel Zeit bis zur Kundenpräsentation / Inbetriebnahme schadet
 - ◆ kein Feedback aus laufendem Betrieb
 - ◆ ... das ist aber das wertvollste Feedback überhaupt
- wenig Einflussmöglichkeiten durch Programmier-Team
 - ◆ Zeitrahmen ist meist vom Kunden bestimmt
 - ⇒ Jahr 2000-Problem, Euro-Umstellung, nächste Messe, ...

Einfluss der Qualität

- externe Qualität
 - ◆ was der Kunde sieht: Funktionalität, Geschwindigkeit, Zuverlässigkeit, ...
- interne Qualität
 - ◆ was der Entwickler sieht: Code-Struktur, Wartbarkeit, Verständlichkeit, ...
- geringere interne Qualität
 - ◆ kurzfristige Zeitersparnis
 - ◆ langfristige Wartbarkeitskatastrophe
 - ◆ langfristig schlechte externe Qualität
 - ◆ demoralisierender Effekt im Team
- hohe interne Qualität
 - ◆ langfristig schnellere Entwicklung
 - ◆ dauerhaft gute externe Qualität
 - ◆ bessere Motivation / Zufriedenheit des Teams
 - ◆ Kundenzufriedenheit
- wenig Spielraum!

Einfluss des Funktions-Umfangs

- Wichtigste Einflussmöglichkeit (laut Kent Beck)
- Idee
 - ◆ Kosten, Zeit und Qualität festlegen
 - ◆ realisierbaren Funktionsumfang ermitteln
- Vorteile
 - ◆ leichte Anpassbarkeit an Änderungsanforderungen
- Risiko
 - ◆ zu viele / zu wichtige Funktionen werden gestrichen
- XP-Ansatz
 - ◆ Wichtigste Kundenanforderungen zuerst realisieren
 - ⇒ nur Funktionalität mit geringer Priorität wird evtl. nicht realisiert
 - ◆ Aufwandsschätzungen mit Feedback
 - ⇒ bessere Schätzungen bedeuten weniger unrealisierbare Dinge

Einflussfaktoren: Fazit / Empfehlungen

- Kosten → Ressourcen bedarfsgerecht einsetzen
- Zeit → keine Einflussmöglichkeit, da extern bestimmt!
(Termine werden eingehalten! Immer!!!)
- Qualität → hohe interne Qualität anstreben!

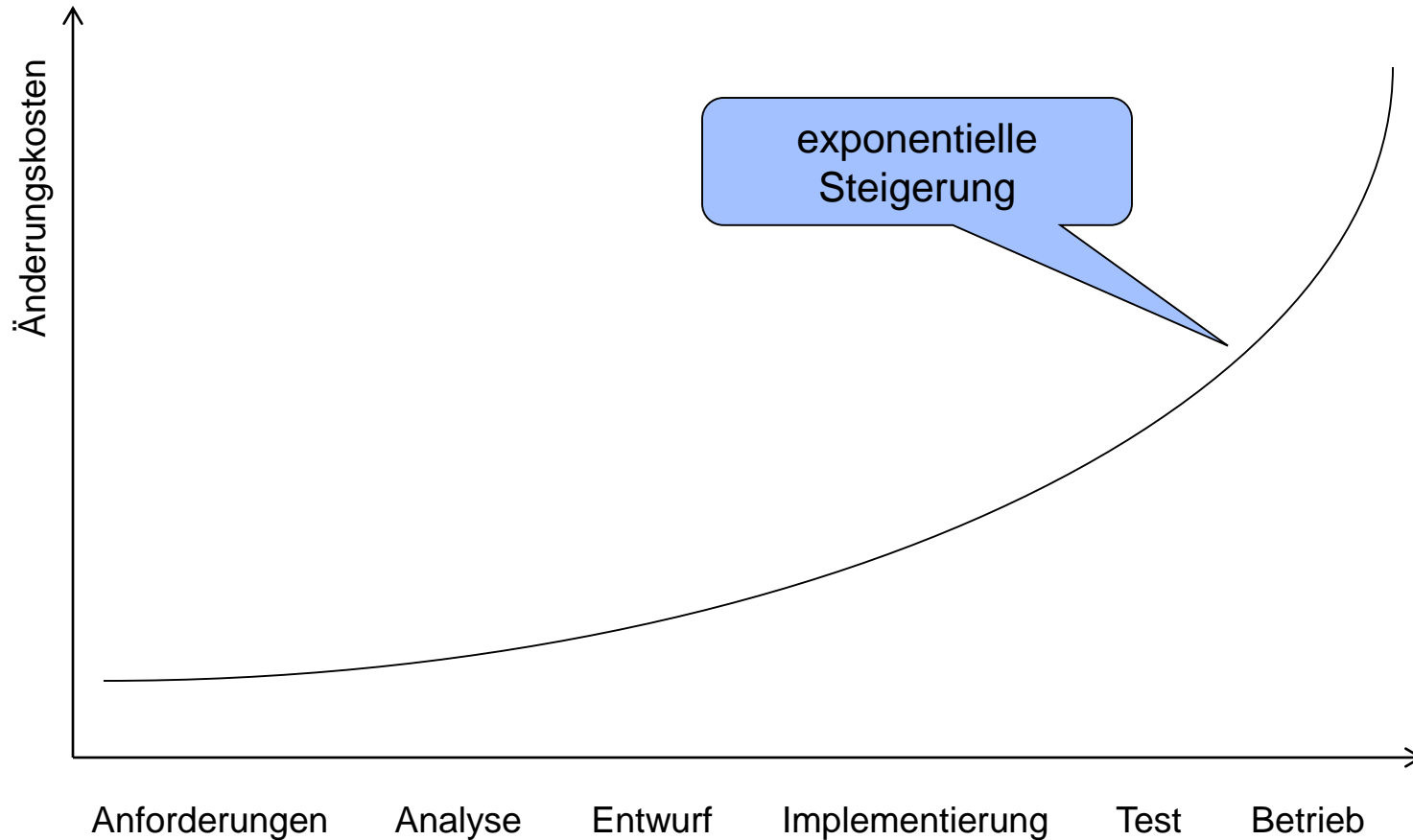
bestimmen ↓

- Umfang → wichtigste Einflussmöglichkeit
 - Kunde bestimmt, worauf er am ehesten verzichten kann
 - Maximierung des Kundennutzens der trotz evtl. reduziertem Funktions-Umfang im Zeit- und Kostenrahmen in hoher Qualität realisierbar ist

**"So einfach wie möglich" klingt gut –
Aber: Was ist mit nachträglichen
Änderungen?**

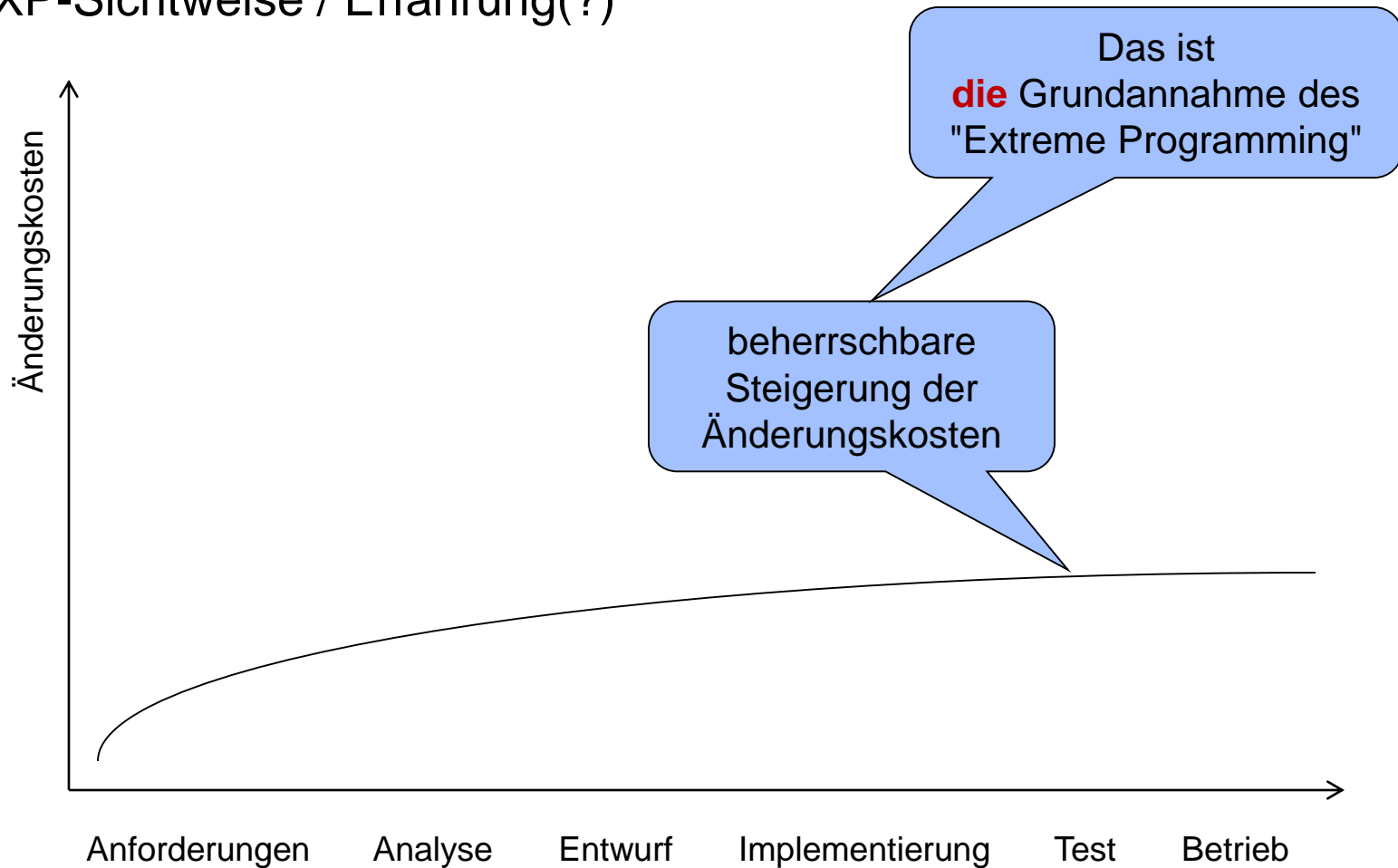
Wie teuer sind Änderungen?

- Traditionelle Sichtweise / Erfahrung



Wie teuer sind Änderungen?

- XP-Sichtweise / Erfahrung(?)



Gegenüberstellung

Traditionelle Methodiken

Extreme Programming

- Annahme

- ◆ exponentielle Kostenexplosion

- ◆ asymptotische Kostensteigerung

- Konsequenzen

- ◆ mögliche Änderungen antizipieren

- ◆ Änderungen erst bedenken wenn gefordert

- ◆ entsprechende Möglichkeiten einbauen

- ◆ nur das notwendigste Implementieren

- ◆ komplexere Software

- ◆ einfachere Software

- ◆ höhere Anfangskosten

- ◆ geringere Anfangskosten

- ◆ langsamerer Anfangsfortschritt

- ◆ schnellerer Projektfortschritt

- ◆ geringere Gesamtkosten!!!

- ◆ geringere Gesamtkosten!!!

- ◆ geringere Gesamtzeit!!!

- ◆ geringere Gesamtzeit!!!

- Fragen

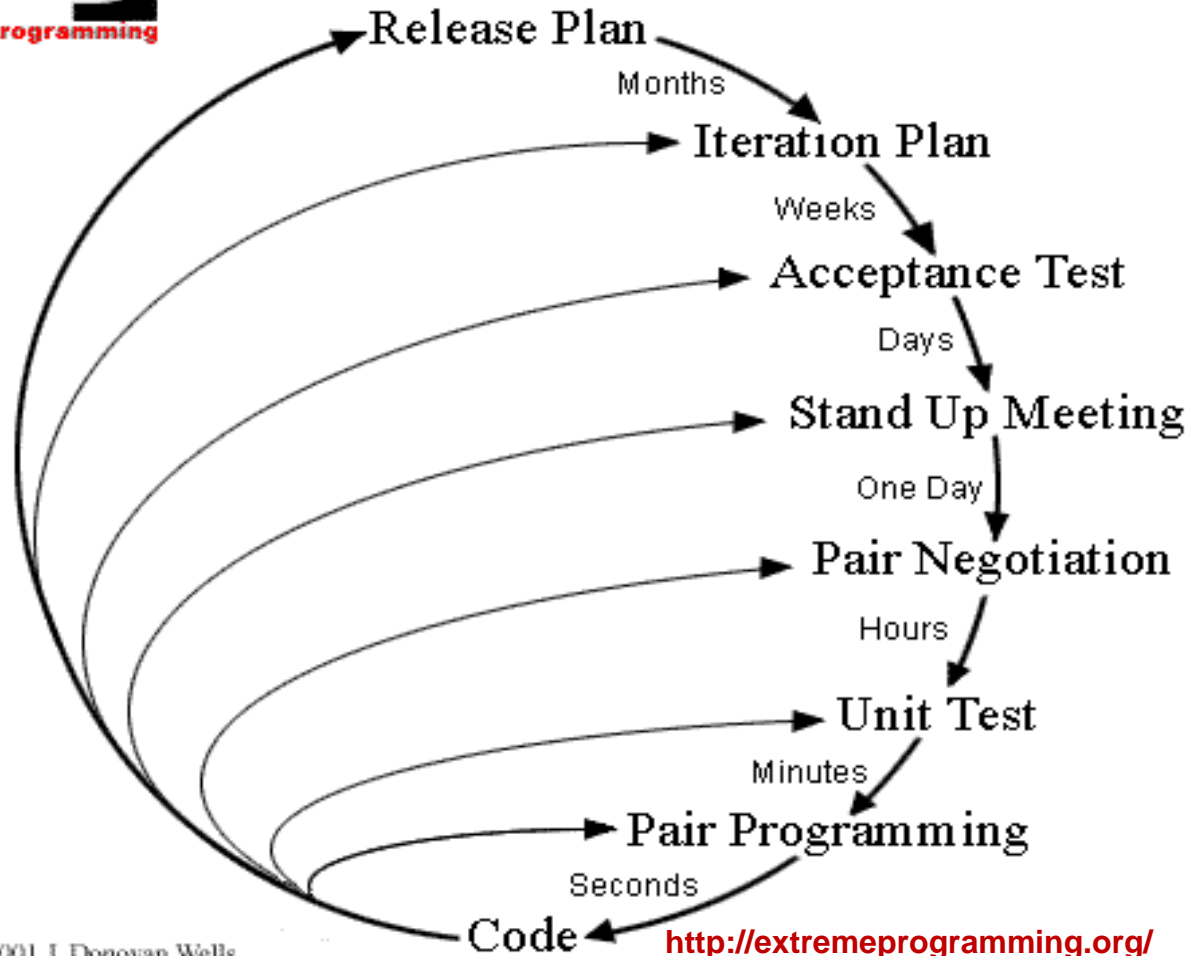
- ◆ welche Annahme stimmt?

- ◆ was würde eine asymptotische Steigerung plausibel machen?

Schnelles Feedback ist das Erfolgsrezept



Planning/Feedback Loops

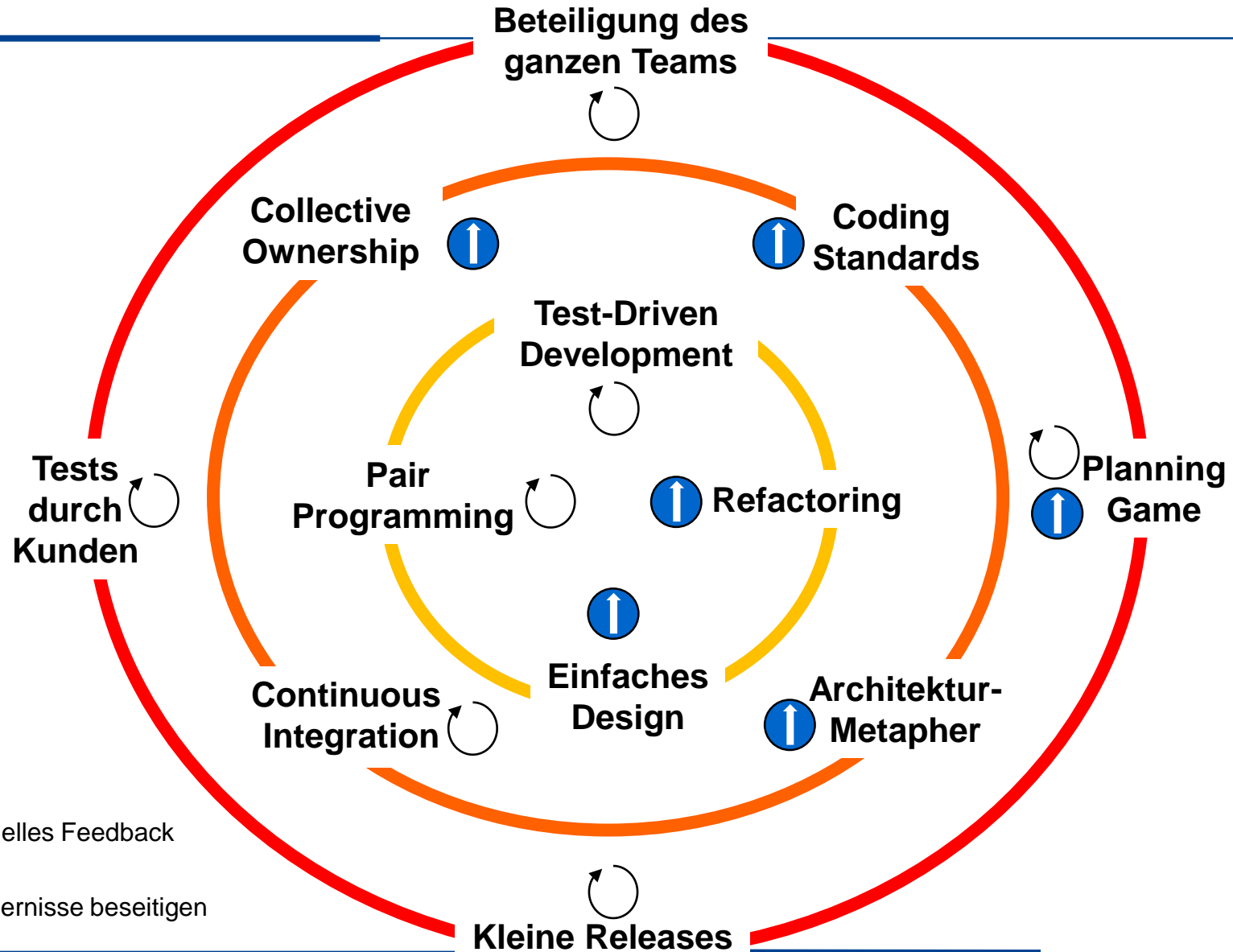


XP is like driving.

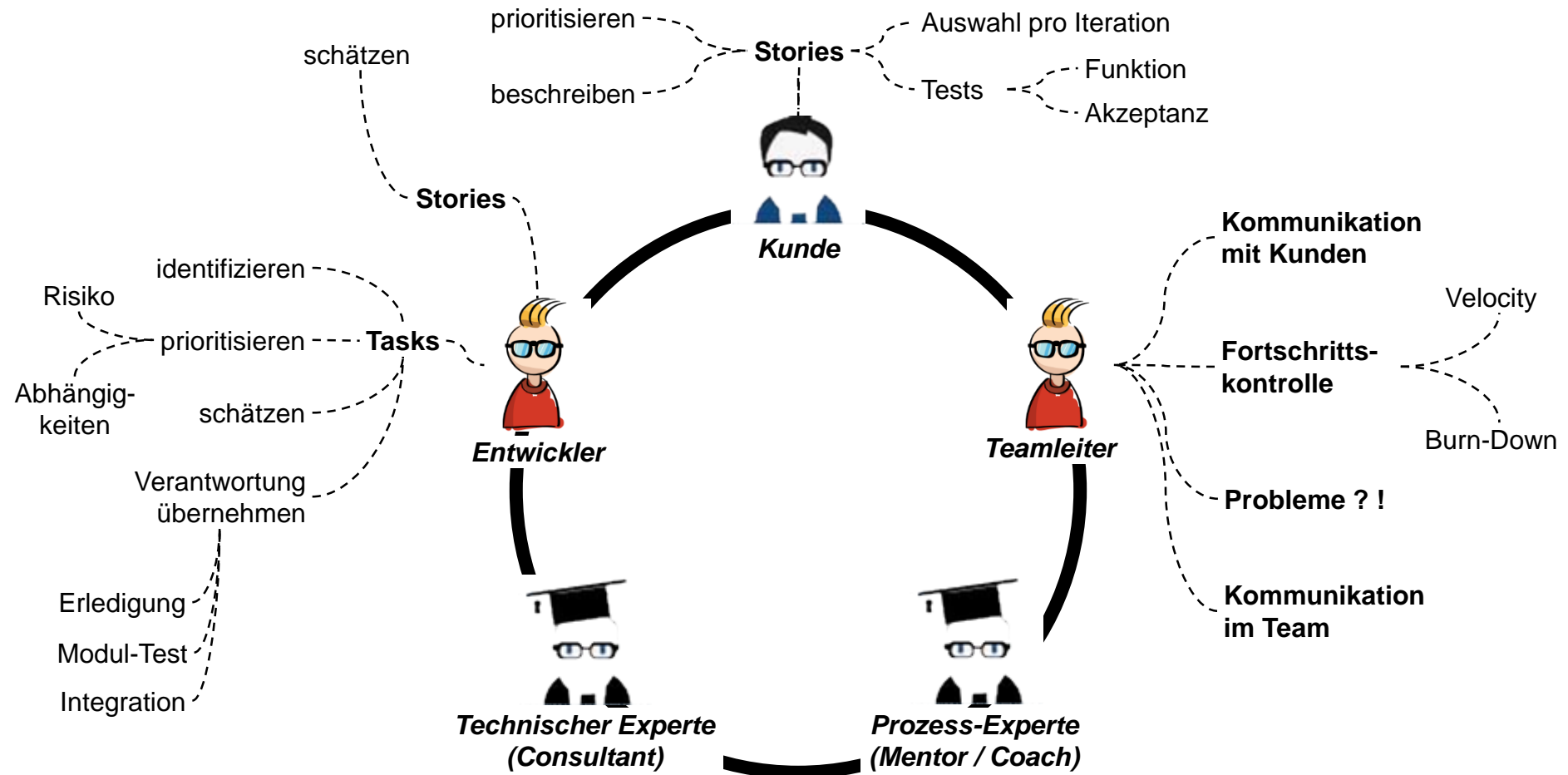
"Driving is about constantly paying attention, making a little correction this way, a little correction that way."

(Kent Beck)

XP Praktiken



XP Rollen



XP Rollen: Kunde

- Spezifiziert
 - ◆ Funktionale und nicht funktionale Anforderungen → „Stories“
 - ◆ Priorisiert die Stories
- Ist verfügbar für klärende Fragen des Teams bezüglich
 - ◆ Unvollständiger, unklarer oder inkonsistenter Anforderungen
 - ◆ Des relevanten Geschäftsprozesses
 - ◆ Des Anwendungsgebietes
- Entscheidet über
 - ◆ Funktionalität (Stories) die in der nächsten Iteration zu implementieren sind

XP Rollen: Kunde (2)

- Führt Systemtests durch
 - ◆ Funktionale Tests („Tut es das was es soll?“)
 - ◆ Acceptance Tests („Tut es dies auf eine Art und Weise, die für Anwender intuitiv und hilfreich ist?“)
- Gibt dem Team Feedback über die getesteten Releases

XP Rollen: Entwickler (inkl. Teamleiter)

- Führen durch
 - ◆ Schätzungen der Zeit / Schwierigkeit und des Risikos der Stories
 - ◆ Unterteilung der Stories in Tasks
 - ◆ Schätzung der Task - Implementationsdauer
 - ◆ Priorisieren die Tasks
- Verpflichten sich
 - ◆ eine Task auszuführen
 - ◆ Kollegen zu helfen wenn nötig
- Entscheiden über
 - ◆ Technische Folgen der Kundenanforderungen (z.B. Wahl eines DBMS)
 - ◆ Den Prozess (wie das Team arbeitet und sich selbst organisiert)
 - ◆ Internes Zeitmanagement
 - ⇒ Was passiert während einer Iteration zuerst?
 - ⇒ Erledige die Tasks mit dem höchsten Risiko zuerst!

XP Rollen: Teamleiter

- **Kontrolliert den Prozess**
 - ◆ Leitet das Team durch den täglichen Ablauf
 - ◆ Behält den Fortschritt des Teams im Auge
 - ⇒ Vergleicht ihn mit den Schätzungen (z.B. mit „Burn-Down Charts“)
 - ⇒ Gibt dem Team Feedback über das Verhältnis von Schätzung zu Realität
 - ◆ Identifiziert Probleme oder „Flaschenhalse“ im Voraus
 - ⇒ Denkt über benötigte Änderungen nach (Prozess, Ablauf, Team, ...)

- **Vermittelt die Kommunikation mit dem Kunden**
 - ◆ Erklärt dem Kunden die Folgen seiner Anforderungen
 - ◆ Erklärt dem Kunden auftretende Probleme
 - ◆ Regt Diskussion über notwendige Zurückstellung von Stories in die nächste Iteration an.

XP Rollen: Teamleiter (2)

- Leitet die Diskussionen des Teams und erklärt Diskussionstechniken (wenn nötig)
- Zielgerichtete Diskussion
 - ◆ Konstruktive Vorschläge
 - ◆ Den Kollegen zuhören
 - ◆ Klare Ergebnisse
 - ◆ Verpflichtungen: Was muss wann von wem getan werden?

XP Rollen: Technischer Experte („Consultant“)

- Ist Experte auf einem für das Projekt wichtigen Feld
- Ist in der Lage seine Expertise dem Team zu vermitteln
 - ◆ Präsentationen
 - ◆ Tutorials
 - ◆ Pair Programming mit Teammitgliedern
 - ◆ Beobachten und beraten von eigenständig arbeitenden Teammitgliedern
 - ◆ Macht Code Reviews
- Muss verfügbar sein wenn er benötigt wird
 - ◆ Gut wenn er dauerhaft vor Ort ist (aber nicht nötig)
 - ◆ Ausreichend wenn er sich in der Nähe aufhält und für das Team verfügbar ist wenn er gebraucht wird

XP Rollen: XP Mentor

- Hat Erfahrung in XP Techniken und Praktiken
- Kann das Team durch den XP Prozess führen
 - ◆ Erklärt XP Techniken und Praktiken
 - ◆ Erklärt ihren Wert und die Auswirkungen wenn man ihnen nicht folgt
 - ◆ Überwacht den Prozess und zeigt Techniken auf die
 - ⇒ nicht angewendet werden
 - ⇒ nicht effektiv angewendet werden
 - ⇒ falsch angewendet werden
 - ◆ Überwacht den Prozess und macht Aufmerksam auf
 - ⇒ nicht-XP Elemente
 - ⇒ anti-XP Elementeund erklärt ihre Gefahren sowie XP-Style Alternativen

XP Rollen: XP Mentor (2)

- Mentor ist Coach (Betreuer) des Teams
 - ◆ "Ein guter Lehrer macht sich selbst langfristig überflüssig".
 - ◆ Entscheidet nicht, sondern
 - ◆ ... hilft anderen gute Entscheidungen zu treffen
 - ◆ Implementiert selbst wenig sondern
 - ◆ ... ist als "pair programmer" für Neulinge verfügbar
 - ◆ ... sieht langfristige Refactorings voraus und ermutigt dahingehend
 - ◆ ... hilft in speziellen technischen Bereichen
 - ◆ Erklärt den Prozess auch dem Management

XP-Management

- Metriken

- ◆ Liefern Feedback über Projektfortschritt

- ⇒ Besseres Verständnis des Prozesses
 - ⇒ Frühzeitige Identifikation von Problemen

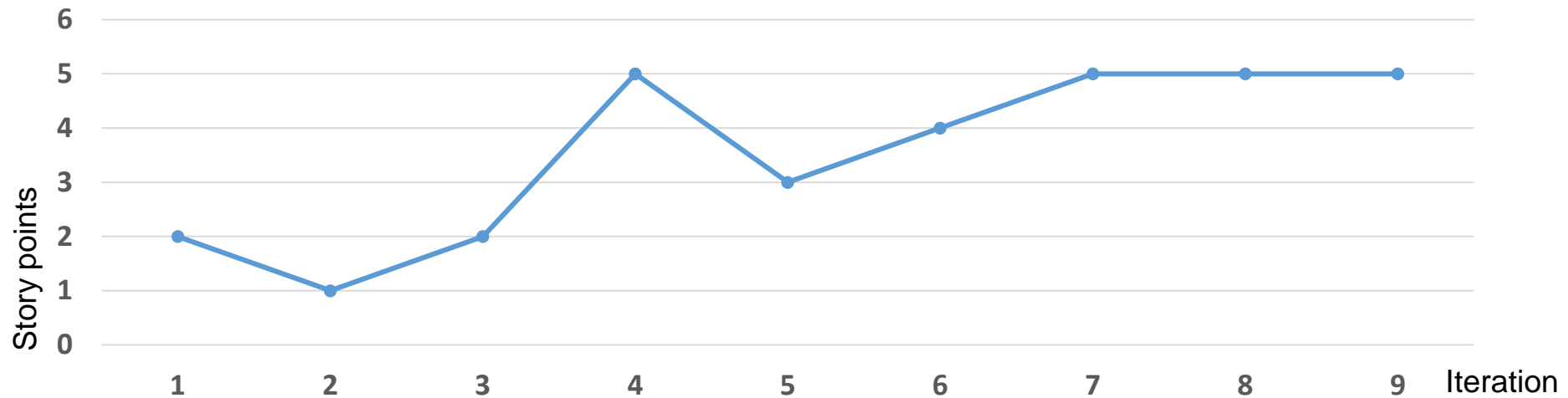
- ◆ „Big Visible Chart“


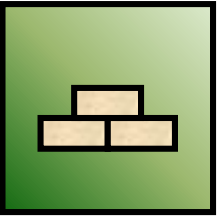
- ⇒ „Project Velocity“ = „Geschwindigkeit“ = Geschaffter Aufwand / Zeit
 - ⇒ „Burndown Chart“ = Verbleibender Aufwand bis Iterationsende

- Eingriff: Wenn's sein muss auch unpopuläre Korrekturen hinsichtlich

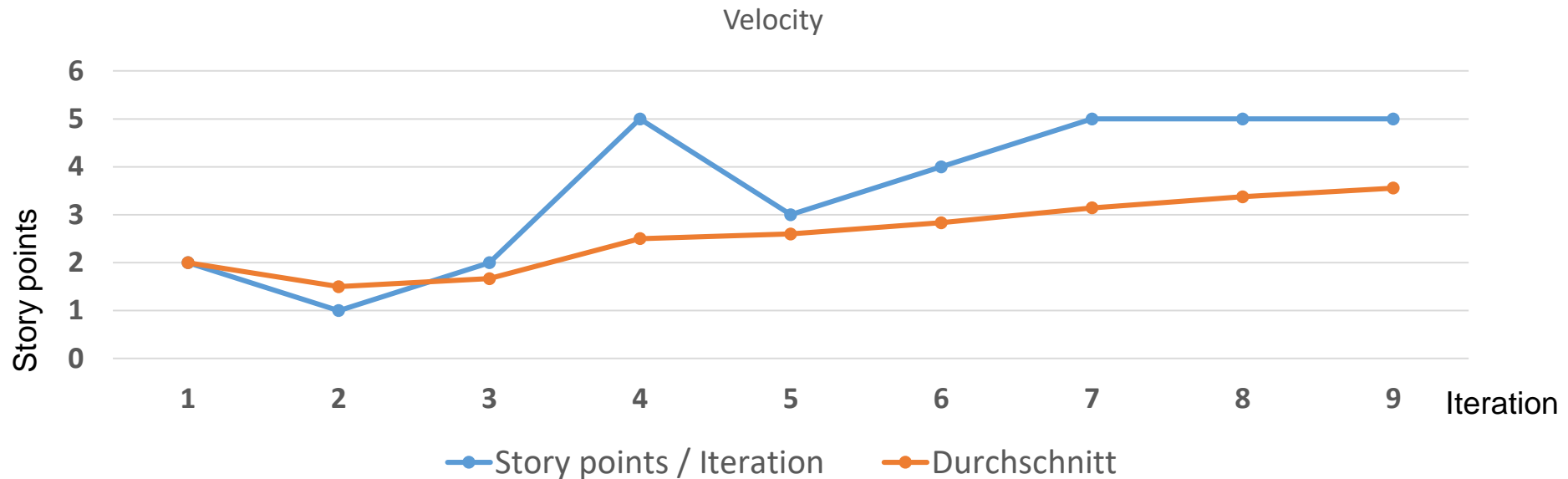
1. Funktionsumfang
2. Architektur
3. Team-Zusammensetzung

Project Velocity ► Definition



$$\text{Geschwindigkeit} = \frac{\text{Strecke}}{\text{Zeit}} = \frac{\text{„Story points“}}{\text{Iteration oder Tag}}$$


Project Velocity ► Nutzen



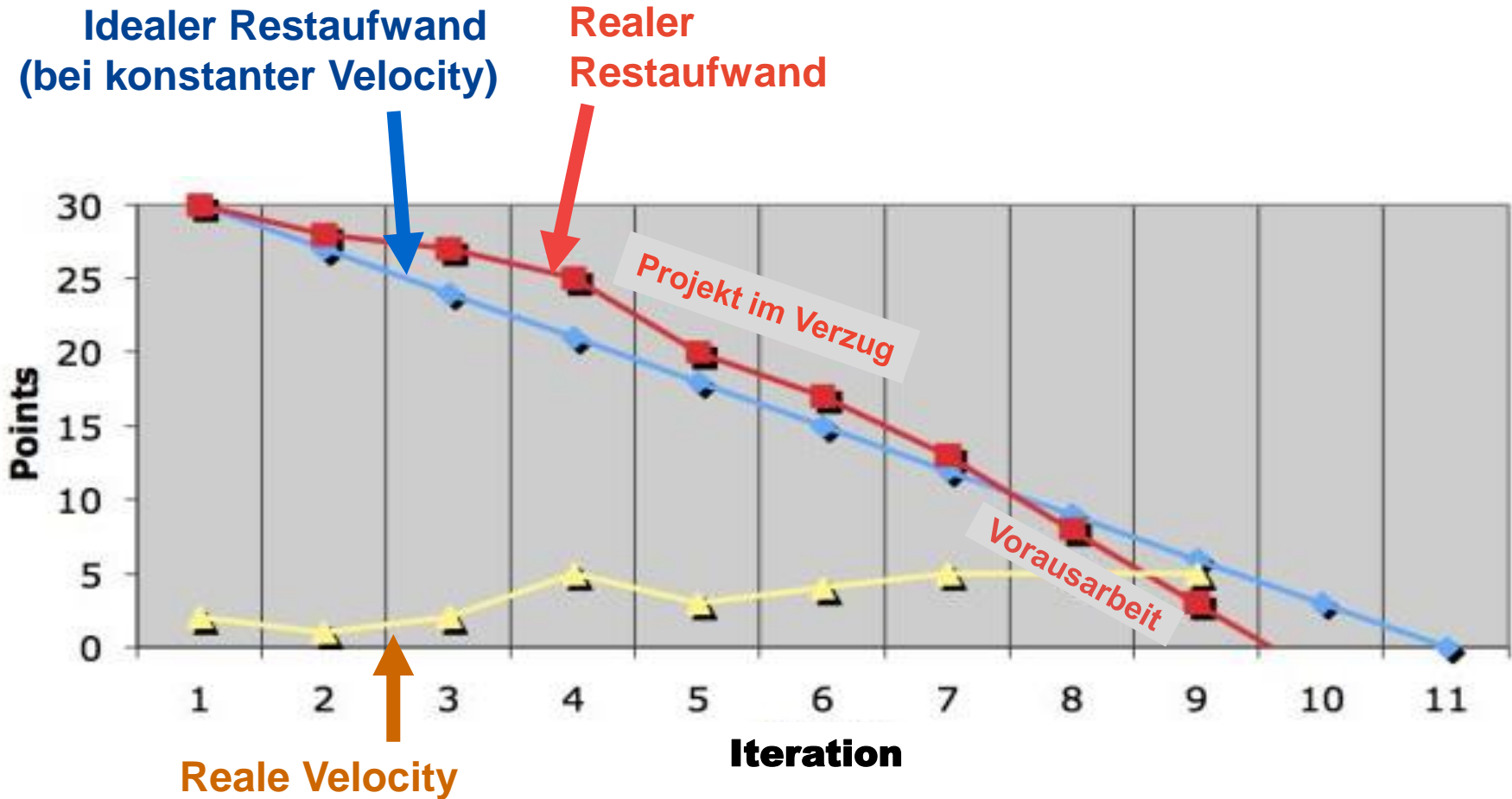
Eigene Durchschnittsgeschwindigkeit bestimmen

- Trend
 - ◆ Im Beispiel: Zunehmend 😊
- Vorhersage
 - ◆ Bei Durchschnittsgeschwindigkeit von 4 Story Points pro Iteration braucht ein Projekt mit geschätzten 30 Story Points ca. 8 Iterationen

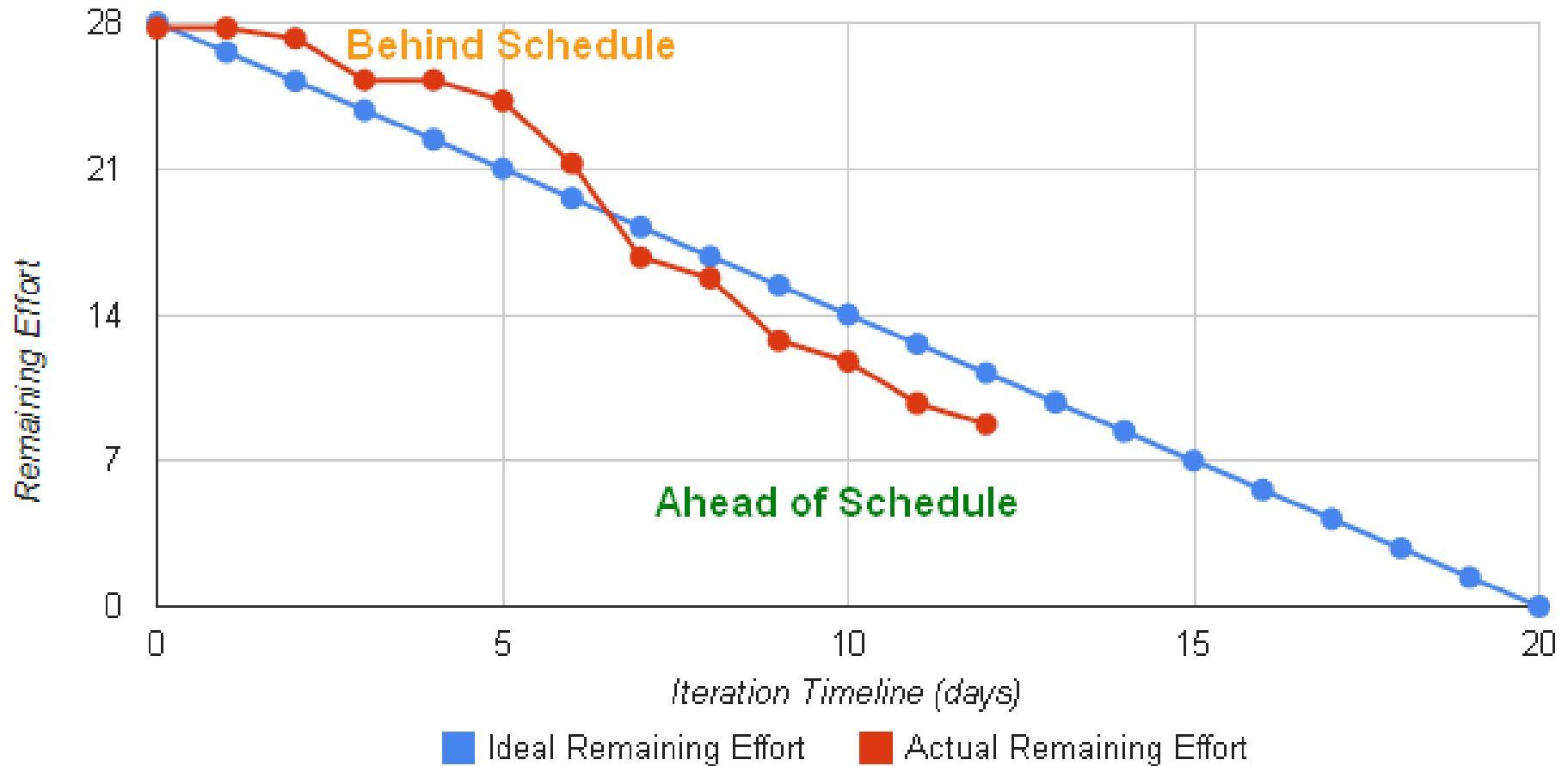
Burn Down Chart ► Idee

- Problem: Velocity ist nicht konstant
 - ◆ Vorhersage schwierig
 - ◆ Wie weiß ich, ob ich noch im Zeitplan bin?
- Idee: „Burn-Down Chart“ = Restaufwand sichtbar machen
 - ◆ Ideal versus Real
 - ◆ Unterschied zeigt Verzug oder Vorarbeit an

Burn Down Chart ▶ Beispiel 1



Burn Down Chart ▶ Beispiel 2



Zusammenfassung

Schnelles Feedback

- ◆ Kunde im Team
- ◆ Pair Programming
- ◆ kurze Iterationen
- ◆ Testing, ...

Laufende Prozessverbesserung

- ◆ Laufende Kontrolle
(Velocity, Burn-down)
- ◆ Rückblick und Reflektion
nach jeder Iteration

Beseitigung von Reibungsverlust

- ◆ Refactoring
- ◆ Collective code ownership
- ◆ Coding standards
- ◆ Effektive Diskussionen (Planning Game, Stand- up meetings)
- ◆ Architektur-Metapher

Motivation

- ◆ Einbeziehung aller Teilnehmer
- ◆ Entwickler schätzen Aufwand
- ◆ Entwickler wählen Tasks
- ◆ Entwickler wählen Partner

