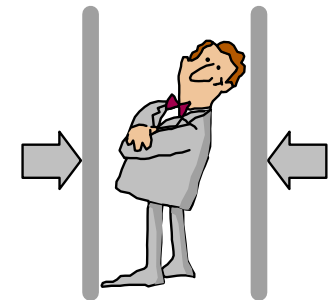
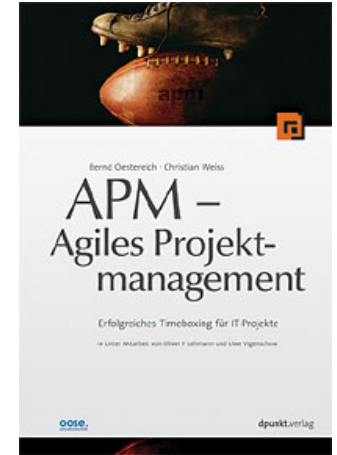




Christian Weiss
(Geschäftsführer)

christian.weiss@oose.de

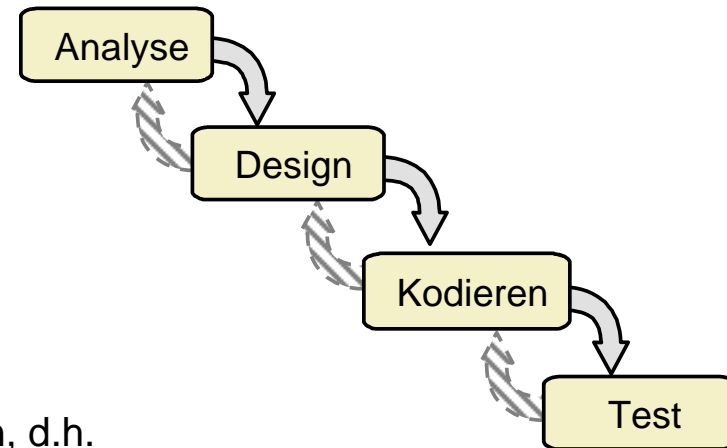
Iteratives Vorgehen und Timeboxing



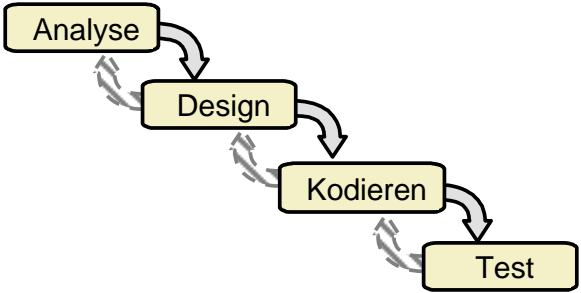
Was kennzeichnet ein reinrassiges Wasserfallmodell ?

Der Entwicklungsprozess ist in klar umrissene Phasen und Ergebnisse unterteilt:

- Analyse** Anforderungen komplett erheben und abstimmen
- Design** Vollständiges Lösungskonzept erarbeiten, d.h. wie können Anforderungen in gegebener Ziel- und Entwicklungsumgebung umgesetzt werden?
- Kodieren** Umsetzung der Anforderungen entsprechend Lösungskonzept
- Test** Test der Lösung gegen die Anforderungen



Bekannte Mängel des Wasserfallmodells

- Anforderungen müssen anfangs weitgehend vollständig bekannt und sehr detailliert sein.
 - Anforderungen müssen stabil bleiben.
 - Erkenntnisse während Entwicklung erfordern Rücksprung in vorige Phase. Problematisch, da keine phasendurchgängige Methodik vorhanden.
 - Großer Abstand zwischen Analyse und Realisierung = dokumentenlastige Konser-vierung von Anforderungen/Entscheidungen.
 - Endlich fertiggestellte Analysen werden hinfällig wegen zwischenzeitlich geänderter Anforderungen.
 - Beteiligte verschiedener Disziplinen spielen lange „Stille Post“ über Papierberge.
 - Mitarbeiter verschiedener Disziplinen sind ungleichmäßig belastet.
 - Big-Bang-Integration bringt neue Show-Stopper – leider erst zum Schluss.
- 
- Das Diagramm zeigt das Wasserfallmodell als eine Abfolge von vier Phasen: Analyse, Design, Kodieren und Test. Die Phasen sind in gelben Kästen dargestellt und durch Pfeile verbunden, die von oben links nach unten rechts zeigen. Zwischen den Phasen befinden sich gestrichelte Rückkopplungspfeile, die nach oben zeigen, was die Möglichkeit des Rücksprungs in eine frühere Phase darstellt.
- Projektfortschritt schwer abschätzbar, lange Zeit sind nur Stunden-verbrauchsmessungen möglich. Fortschrittsaussagen sind trügerisch.
 - Mentalität: einmal planen, dann nach Plan durchführen, Korrektur = Fehler
 - Folgen: Defizite, Verzögerungen, Probleme werden verschleiert, schöngeredet, Salamtaktik
 - Tatsache: Projekte sind nur begrenzt planbar, Realität überholt den Plan, regelmäßige Planungskorrektur ist unvermeidlich.

Das Wasserfallmodell – ein historisches Missverständnis.

Der Wasserfallprozess-„Erfinder“ Winston Royce beschrieb es als

- ein sehr einfaches Modell
- ohne empirische Daten
- ausdrücklich nur für einfachste Projekte

Quelle: Winston Royce, *Managing the development of large software systems*, Proceedings of IEEE Westcon, 1970



„Mein Vater war immer ein Vertreter von iterativen, inkrementellen und evolutionären Entwicklungsmodellen. Sein Arbeitspapier beschrieb das Wasserfallmodell als die einfachste Möglichkeit, welches aber nur für die unkompliziertesten Projekte funktioniert.“ (Walker Royce)



Der US-Standard DoD-STD-2167 favorisierte das Wasserfallmodell, andere Standards (CMMI, V-Modell) zogen nach.

Dabei war David Maibor (Autor DOD-STD-2167) mit timebox-getriebener iterativer Entwicklung und evolutionären Anforderungen nicht vertraut und im Nachhinein hätte er dies viel deutlicher empfohlen als im STD-2167!

Quelle: Craig Larman, *The historical accident of waterfall validity*, in: Agile & iterative development, 2004

Seit 30 Jahren bewährt: Iterativ-inkrementelle Entwicklung



1977 – 1980: IBM FSD entwickelt die NASA Space-Shuttle-Software in 17 Iterationen über 31 Monate mit durchschnittlich 8 Wochen/Iteration.

Quelle: Madden/Rone: *Design, Development, Integration: Space Shuttle Flight Software*. Communications of the ACM, 11/ 1984.



1985 – Barry Boehm publiziert das Spiralmodell

Quelle: Barry Boehm: *A Spiral Model of Software Development and Enhancement*. Proceedings International Workshop on Software Process and Software Environments. 3/1985.

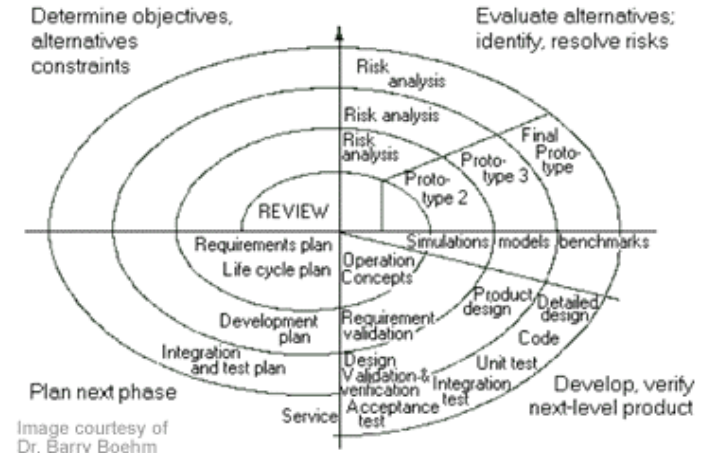


Image courtesy of
Dr. Barry Boehm

Standish-Chaos-Report bestätigt iterative Verfahren!

Standish "Chaos"

Failure Up, Success Down, in 2004

Remember Standish, the company that publishes those "Chaos" reports that tend to show that the software crisis is for real? Well, they're at it again.

The Standish 2004 Chaos study shows a decline in project success rates and (of course) an increase in failures. From 2002 to 2004, we have:

- 15% failure rate increasing to 18%
- 34% success rate declining to 29%
- cost overruns increasing from 43% to 56%

- schedule overruns increasing from 82% to 84%
- features/functions requirements met, declining from 67% to 64%

Why do things seem to be getting worse? Standish says it's because those 2004 projects were bigger and more expensive. They also cited lack of user involvement and project executive support. The study went on to report that the

top success factors for IT projects are

- user involvement
- executive support

- clear business objective
- experienced project manager
- minimal scope and requirements
- an iterative and agile development process(!)
- standard tools and infrastructure
- skilled resources in place
- a formal methodology and financial management

This analysis of the 2004 Chaos findings is based on data from the publication e-Wise Solutions, Dec. 21, 2004.



THE SOFTWARE PRACTITIONER

MARCH – APRIL 2005

The newsletter by and for software professionals.

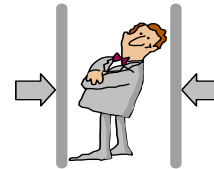
VOLUME 15, NUMBER 2

Phase, Meilenstein, Timebox

Eine **Phase** ist ein zeitlich bzw. sachlogischer Gliederungsabschnitt eines Projekts. Eine Phase fasst eine Menge von Aktivitäten und Ergebnissen zu einer Planungs- und Kontrolleinheit zusammen. Am Ende jeder Phase steht gewöhnlich ein → *Meilenstein*, der die in der Phase zu erzielende Inhalte definiert.

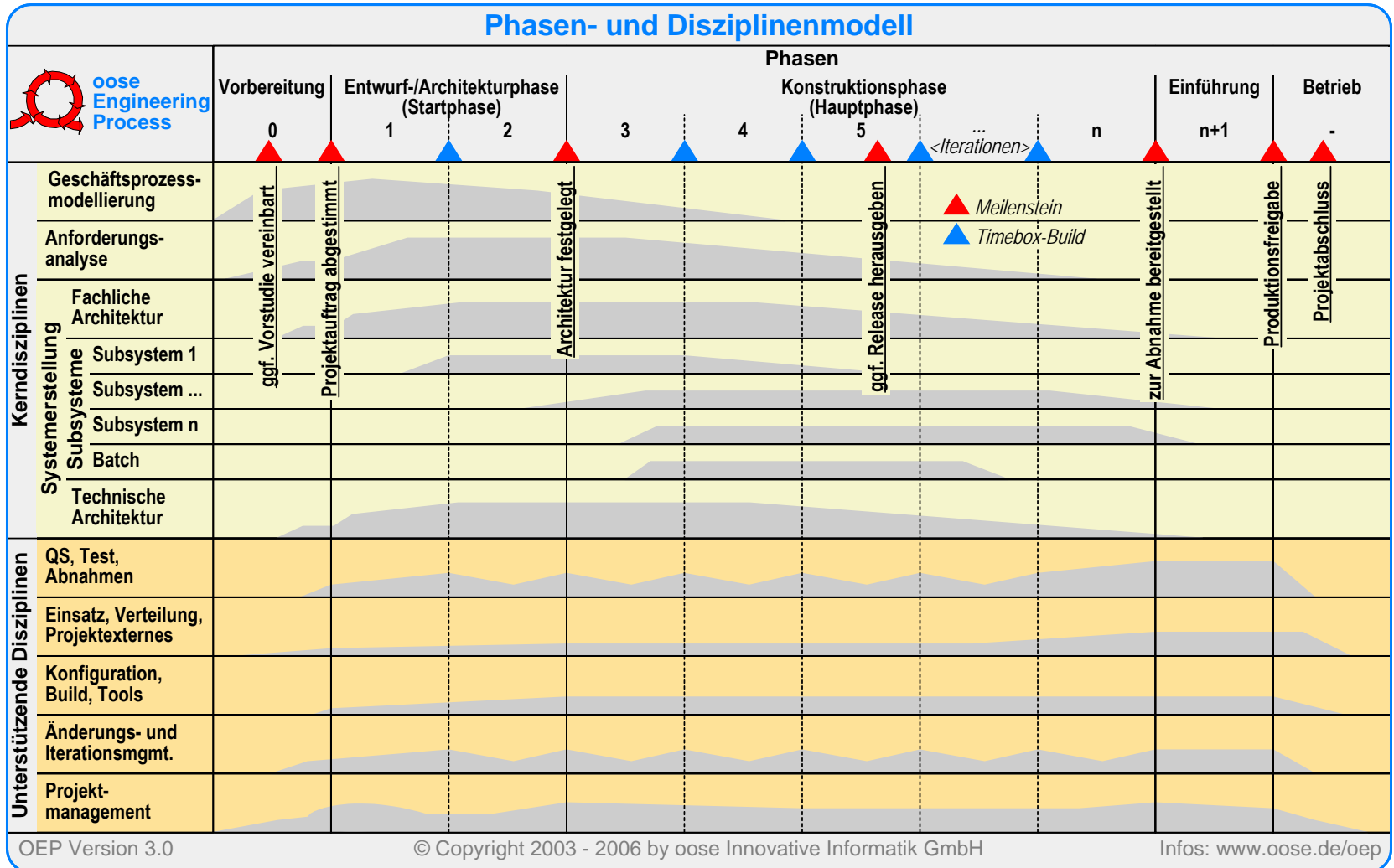


Ein **Meilenstein** (*Milestone*) definiert einen äußerst wichtigen Termin, zu dem eine Menge von Ergebnissen in vorgegebener Detaillierung und Vollständigkeit nachprüfbar und formal dokumentiert vorliegen muss. Liegen die Ergebnisse zum geplanten Termin nicht vor, wird der Meilenstein verschoben.

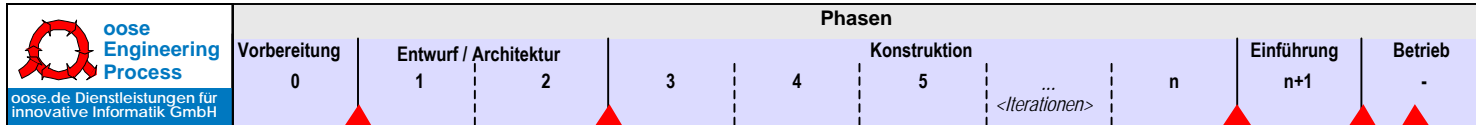


Eine **Timebox** definiert einen unverrückbaren Zeitrahmen, in dem eine Menge von Ergebnissen in vorgegebener Detaillierung und Vollständigkeit vorliegen soll. Liegen die Ergebnisse nicht wie geplant vor, werden die offenen Teile in eine nachfolgende Timebox verschoben. Hierzu werden zum geplanten Endtermin die tatsächlich erreichten Ergebnisse bestimmt.

Überblick Iteratives Vorgehensmodell



Was passiert in welcher Phase?



Vorbereitung: Problem verstehen, Planungs- und Entscheidungsbasis schaffen

- System-Kontext erstellen
- Überblick über Release-Features
- Alle primären Anwendungsfälle finden
- Identifizierung von Risiken und Erfolgskriterien
- Untersuchung von Realisierungsalternativen
- ggf. grobes Fachklassenmodell



Architektur: Lösungsansatz verstehen, Architektur festlegen

- Architekturelevante Anwendungsfälle detaillieren
- Fachliche Architektur (Systempartitionierung)
- Prototypen für hochpriorisierte Anwendungsfälle
- Prototypen für technische Risiken
- Iterations-Features definieren
- Iterationsplan

Konstruktion: Lösung erstellen

- Anwendungsfälle detaillieren
- Design-Modelle erstellen
- Programmieren
- Testen
- Benutzerhandbücher u.v.m.

Einführung: Lösung einführen

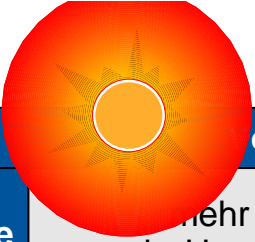
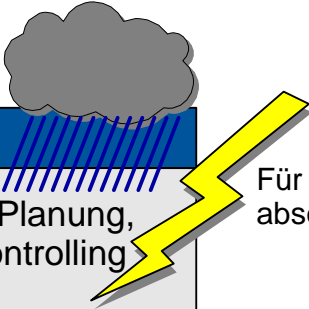
- Test und Abnahme in Zielumgebung
- Schulung
- Rollout
- Inbetriebnahme

Architekturphase: nur ausgesuchte Anforderungen detaillieren.

Design nur für Prototypen. Wichtiges Ergebnis = **Iterationsplan**.

50% des Analyse- und 80% des Design-Aufwands in der Konstruktions-Phase!

Wie lange sollte eine Iteration dauern?

	Vorteile	Nachteile
Kurze Iterationen	mehr Feedback mehr Handlungsoptionen	mehr Aufwand für Planung, Steuerung und Controlling
Lange Iterationen	weniger Aufwand für Planung, Steuerung und Controlling	weniger Feedback abstraktere Ziele weniger Ergebnisorientierung

Für Großprojekte absolut kritisch

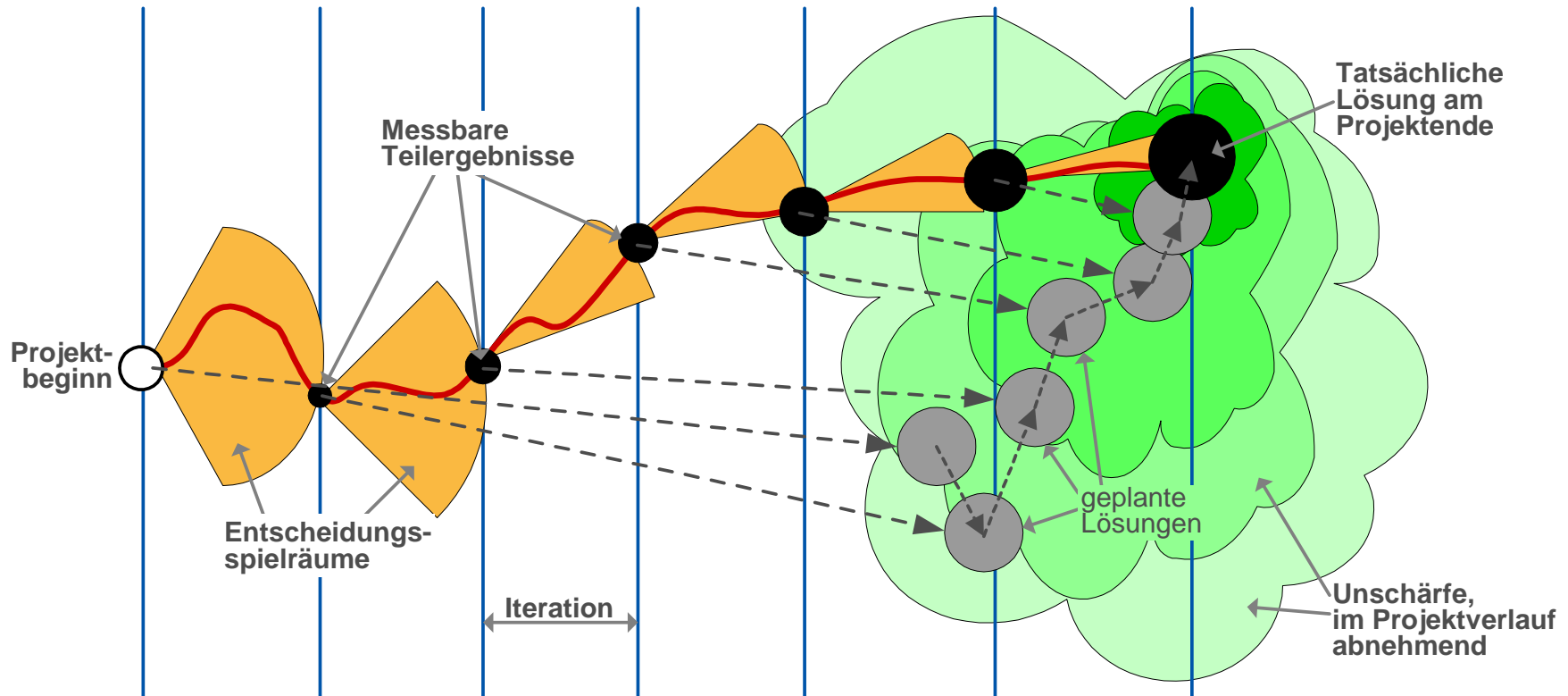
- Iterationsdauer ist abhängig von Teamgröße

Mitarbeiterzahl	Iterationsdauer
4	2 - 4 Wochen
10	3 - 5 Wochen
40	6 - 8 Wochen
>40	6 - 10 Wochen

Vgl. SCRUM:
7 MA -> 6 Wochen (30 Arb.tg.)

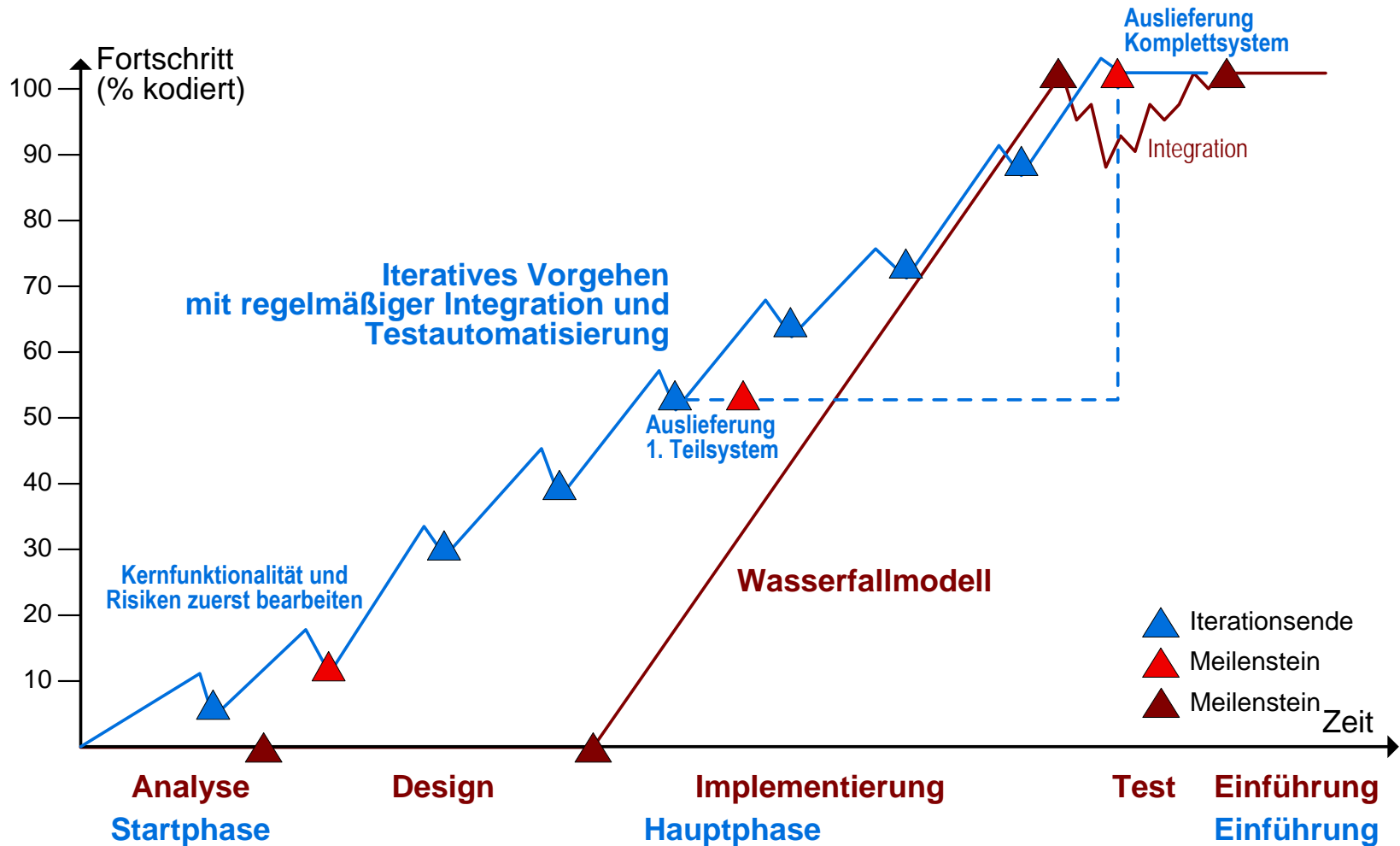
Vgl. Kruchten/RUP:
40 MA -> 12 Wochen
150 MA -> 32 Wochen

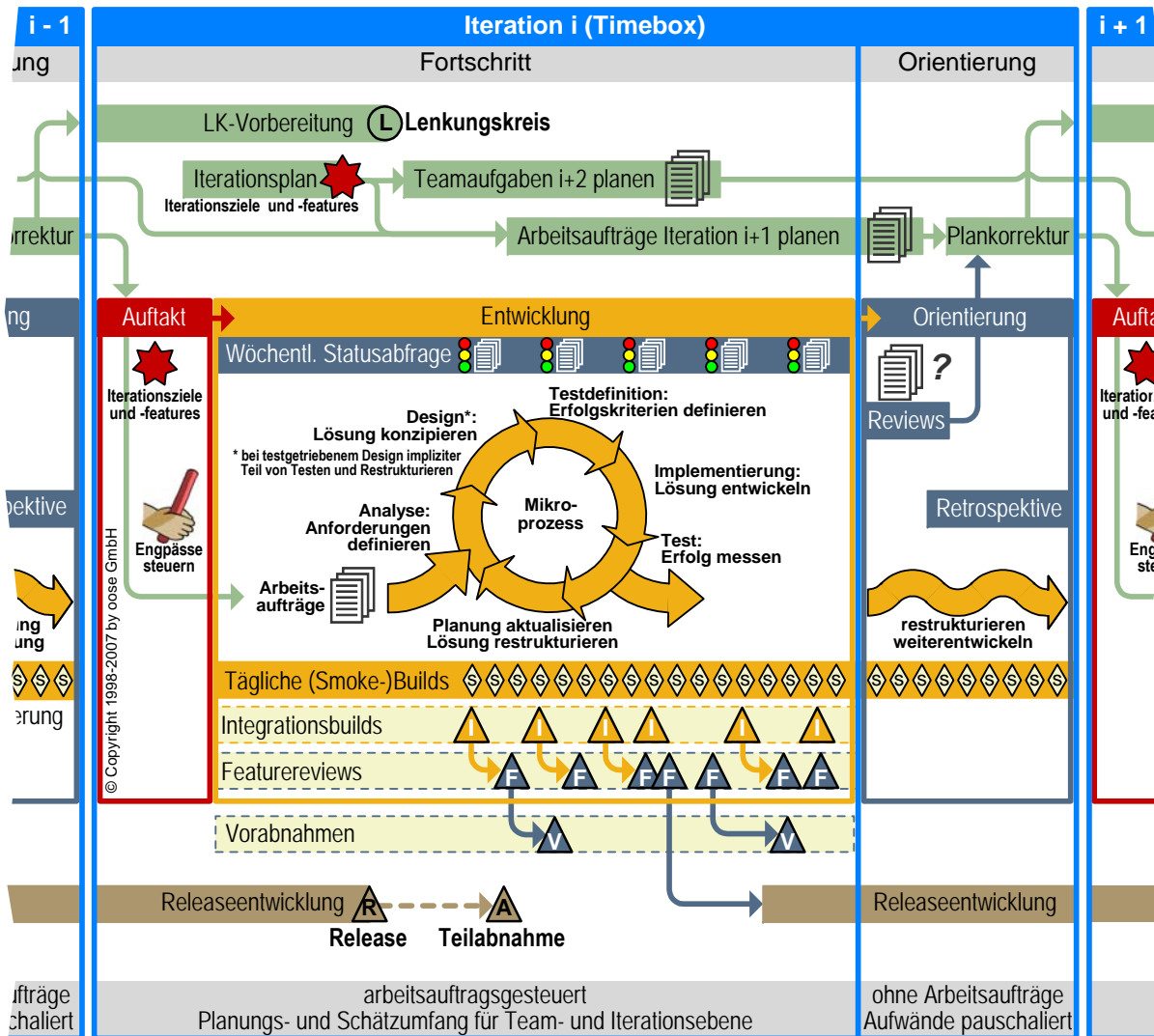
Wolkenmodell



Die Beweglichkeit (Agilität) nimmt (auch) bei iterativem Vorgehen ab

Wasserfallmodell vs. Iteratives Vorgehen





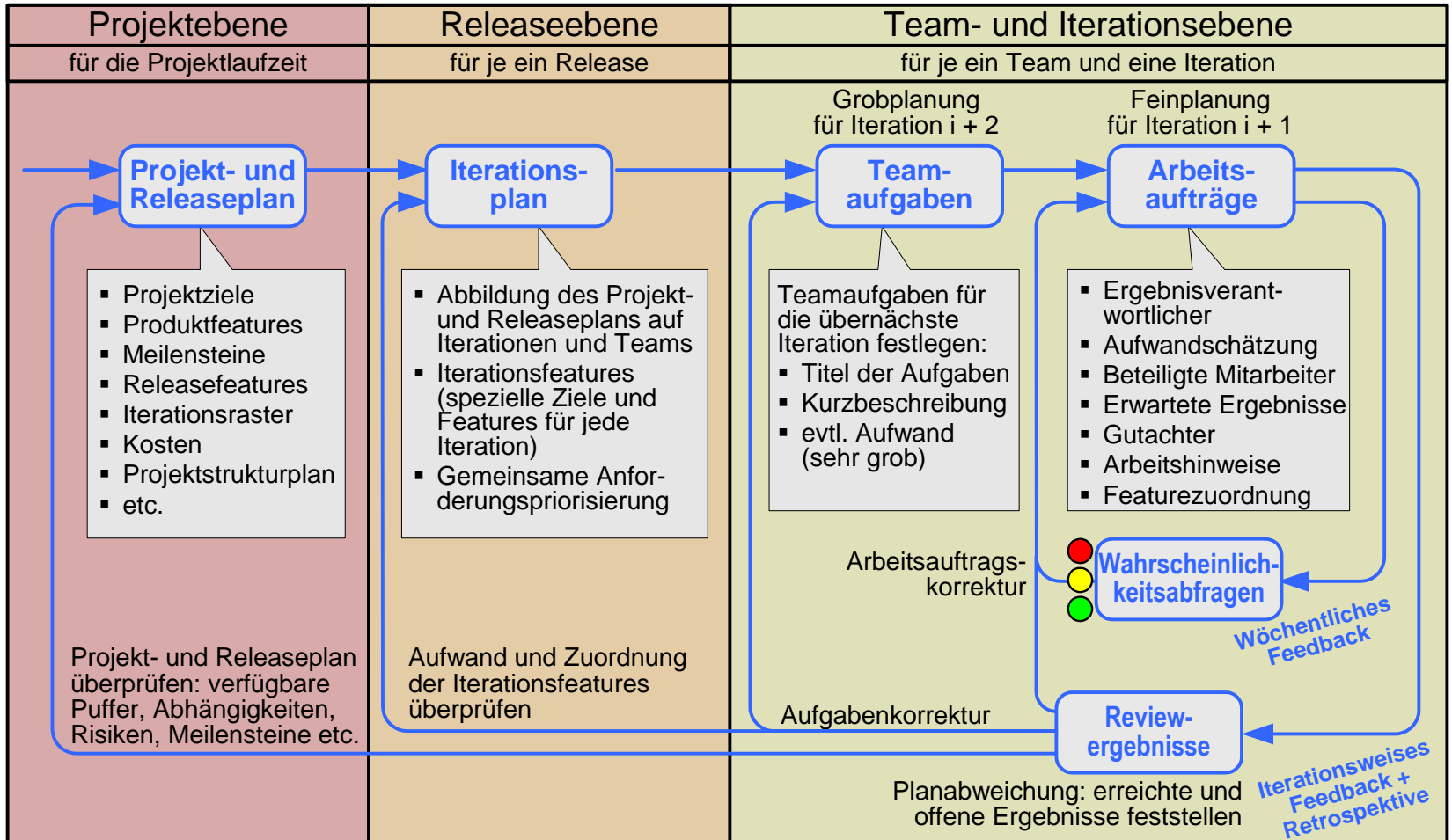
	Ziele setzen
	vorbereiten
	entwickeln
	bewerten
	ausliefern

- Zeitpunkt erreicht (Timebox-Ende)
- Inhalt erfüllt
- Iterationsziele und -features
- Aufgaben bzw. Arbeitsaufträge
- Arbeitsauftragsstatus
- Smoke-Build (täglich oder kontinuierlich)
- Feature-Review
- Vorabnahme auf Basis Integrationsbuild
- Integrationsbuild (wöchentlich oder öfter)
- (Teil-)Abnahme auf Releasebasis
- Fertiges Release

Legende

Orientierung Start Das APM-Timebox-Iterationsmodell Ende Entwicklung Ende Orientierung

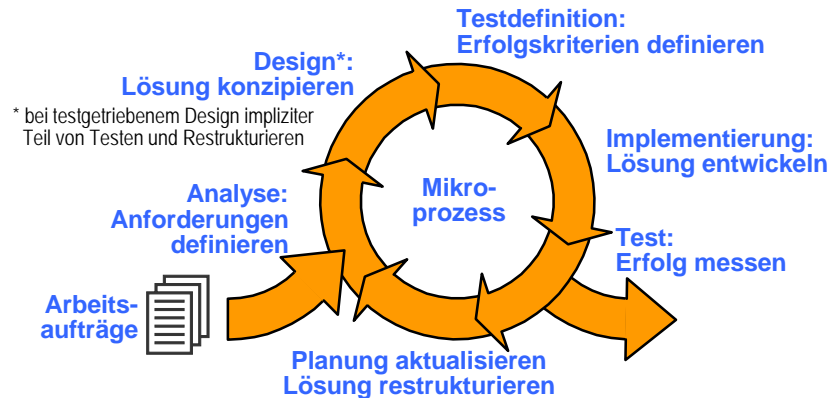
Planungsebenen, -dokumente und Rückkopplungen



Vorteile und Nachteile des Iterativen Vorgehens

Vorteile:

- Besseres Risikomanagement
- Kontinuierliche Integration
- Integrierte Teststrategien
- Flexiblere Auslieferung
- Fortlaufende Planung
- Sichtbarer Projektfortschritt
- Optimierter Entwicklungsprozess
- Geringere Dokumentenlastigkeit
- Höhere Kundenakzeptanz
- Gleichmäßigere Mitarbeiterauslastung
- Motiviertere Mitarbeiter

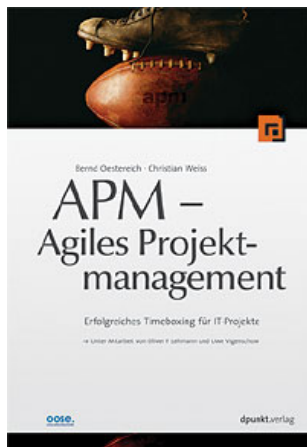


Nachteile:

- Anspruchsvolleres Vorgehen
- Es muss wirklich geplant werden
- Umdenken erforderlich (ggf. Kulturschock)
- Andere sehen besser, was man tut
- Projektstillstand kann nur kurz verheimlicht werden
- Kunde könnte früher aussteigen

Herzlichen Dank!

Mehr zum Thema im APM-Buch oder bei oose:



Businessmanagement

Agile Geschäftsprozesse **BPMx**
Enterprise Architektur **OCEB**
Organisationsentwicklung Pro-
gramm-Management Restrukturierung
SOA Sourcing
Strategien Ver-
änderungspro-
zesse u.v.m.

Projektmanagement

Agilität **Audits** Großprojekte Interims-Management
IPMA **Iteratives Vorgehen** OEP PMI Projekt-Controlling
Risiko-Management **Scrum** Software-Bauleitung
Timeboxing Turnaround-Management Vorgehens-
modelle u.v.m.

Soft Skills

Hochleistungsteams Kommunikation
Konfliktbewältigung **Mediation** Modera-
tion **Systemisches Coaching** Team-
training Veränderungspro-
zesse Verhandlungen u.v.m.

Analyse Architektur

Design
Geschäftsprozesse
Java-Technologien
und -Frameworks

Hardware **Interdisziplinäre**
Systementwicklung Mechanik
Modellbasiertes Engineering Requirements-
Engineering **Spezifikationen** **SysML**
Systemmodellierung V-Modell XT u.v.m.

Systems-Engineering

Methodik **Modellgetriebene Entwicklung**
Programmierung Requirements-Engineering
Testen/QS **UML** **.NET** u.v.m.

Software-Engineering