Softwareentwicklungsprozesse

Softwareentwicklungs-Projekte

- Wie können Anforderungen dokumentiert werden – welche Schwierigkeiten bestehen in der Anwendung und Berücksichtigung einer Anforderungsliste, bzw. um später die vorhandenen Funktionen mit den Forderungen abzugleichen

- Welche Probleme können entstehen, wenn ein Festpreisangebot abgegeben werden soll oder wie kann in klar umrissener Zeit eine Lösung gefunden werden, um das unternehmerische Risiko in den Griff zu bekommen? Wie können verlässliche Aussagen zu Zeit, Kosten und Qualität zu Projektbeginn getroffen werden?

- Was muss getan werden, um ein Softwareentwicklungs-Projekt zu managen? Welche Projektmanagementstrategien können in einem Softwareprojekt Anwendung finden?

Der ideale Mitarbeiter im Projektgeschäft

- Intelligenz
  - Fähigkeit zum Begreifen und Verstehen von Zusammenhängen sowie das Beurteilen derselben
  - Um Probleme zu lösen, braucht es Intelligenz, je mehr davon vorhanden ist, desto besser ist die Problemlösungskompetenz
- Kreativität
  - Kraft, um neue Dinge zu entwickeln, Intuition
  - Das reine Produzieren neuer, ungeordneter Ideen ist im technischen Bereich nicht hilfreich, hier ist es erforderlich, dass neue Ideen mit konkreter Zielstellung kreiert werden
- Entscheidungen treffen
  - Das Erkennen von Zusammenhängen ist erforderlich, um Probleme in kleinere zu zerlegen
  - Wenn Prioritäten erkannt werden, ist dies hilfreich für das Abschätzen notwendiger Zeiten anhand der eingeschlagenen Verfahrensweise
- Gute Problemlöser
  - verfügen über ein gut strukturiertes Wissen
  - können gut vom konkreten zum abstrakten und zurück umschalten
  - können mit unsicheren Fakten umgehen
  - besitzen eine gute Balance zwischen Sturheit und Flexibilität
  - behalten das zu lösende Ziel im Auge

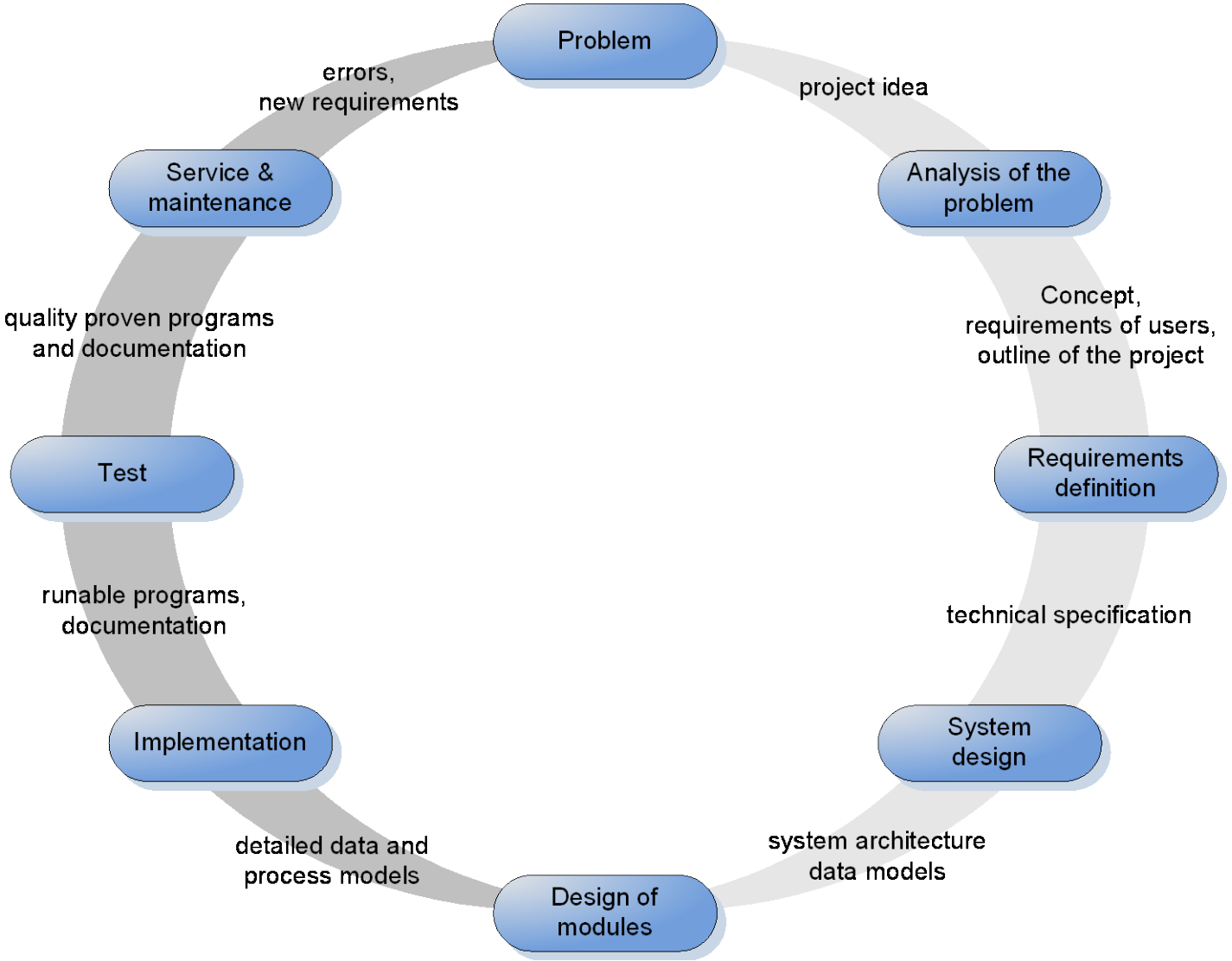
Die objektorientierte Analyse (OOA)

- Beschreibung eines Systems ohne den Hintergrund einer konkreten Programmiersprache

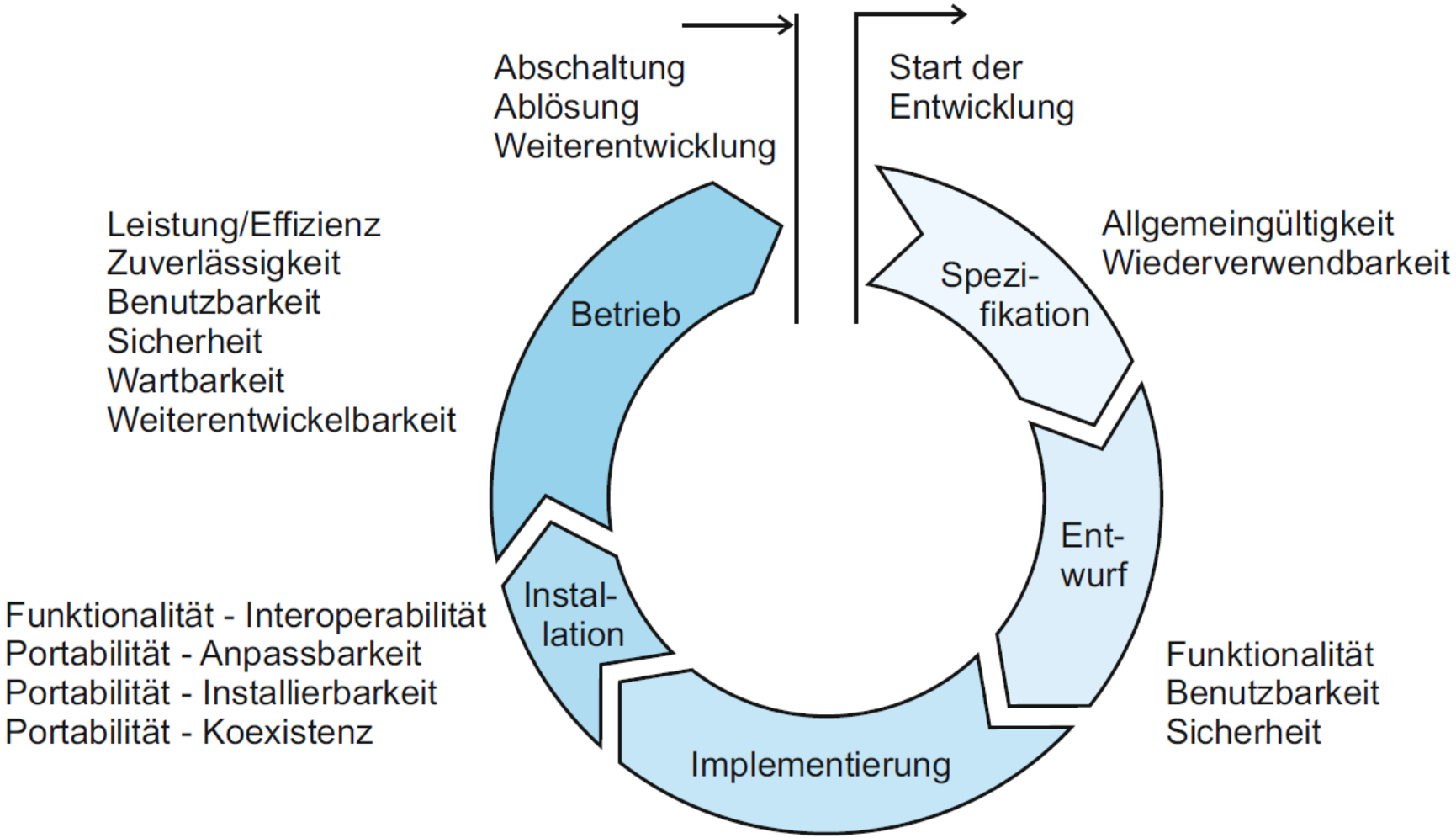
- Ziele
  - Erkennen der grundsätzlichen Zusammenhänge
  - Beschreibung und Definition einer gemeinsamen Problemsicht
  - Diskussionsgrundlage zur Klärung bisher nicht dokumentierter Sachverhalte
  - Zerlegen des Gesamtproblems in kleinere Probleme und deren Zusammenspiel

- Hilfsmittel
  - UML zur Modellierung
  - Use-Case-Diagramme zur Beschreibung ganz allgemeiner Anwendungsfälle ohne Implementierungshintergrund
  - Klassendiagramme zum Aufzeigen der Klassen, Objekte und ihrer Beziehungen
  - Aktivitäten-Diagramme zur Beschreibung von Prozessketten, z.B. zur genaueren Beschreibung eines Anwendungsfalls
  - Weitere Diagrammformen

Der Software - Lebenszyklus

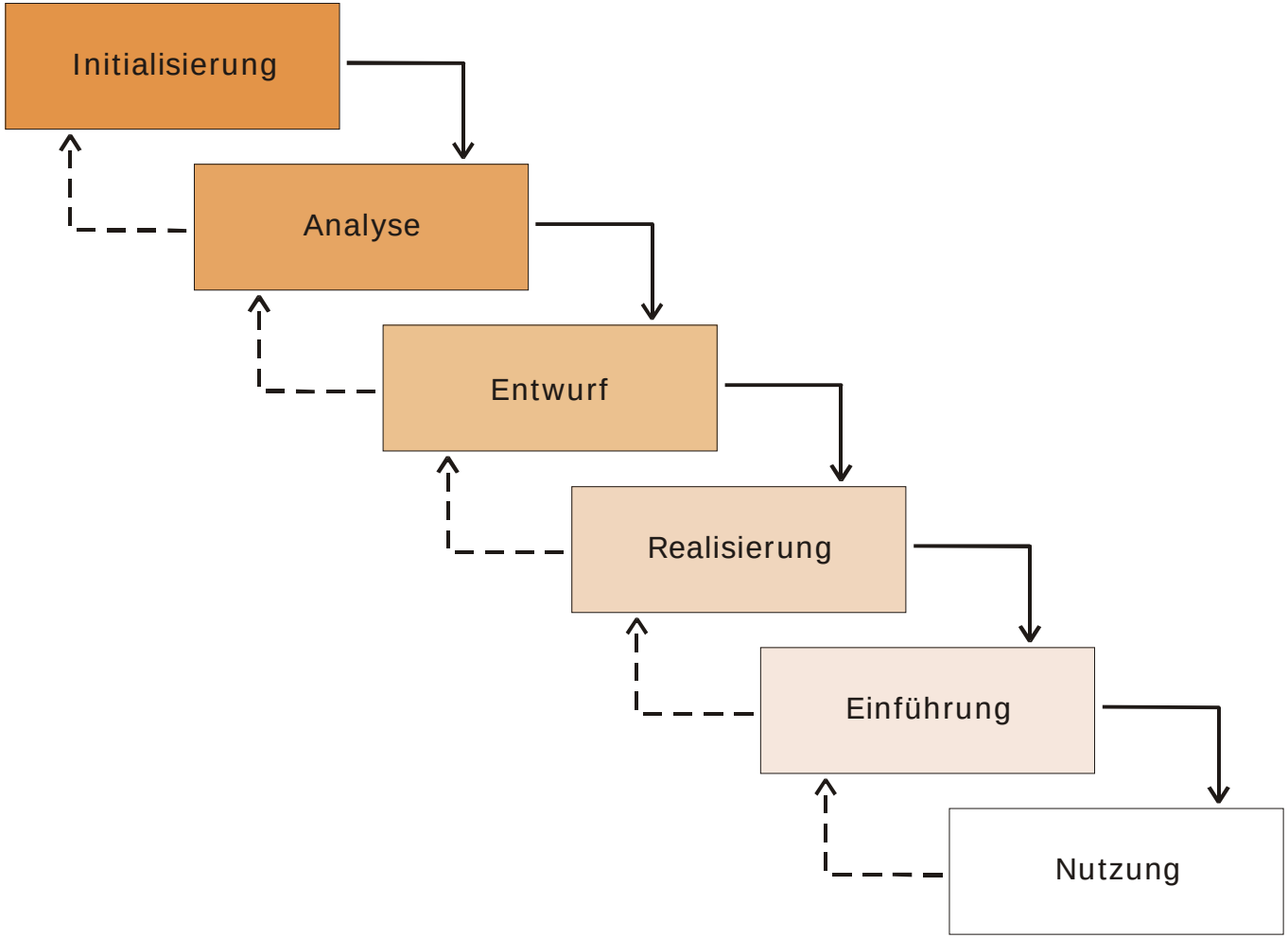


Lebenszyklus und nicht-funktionale Anforderungen

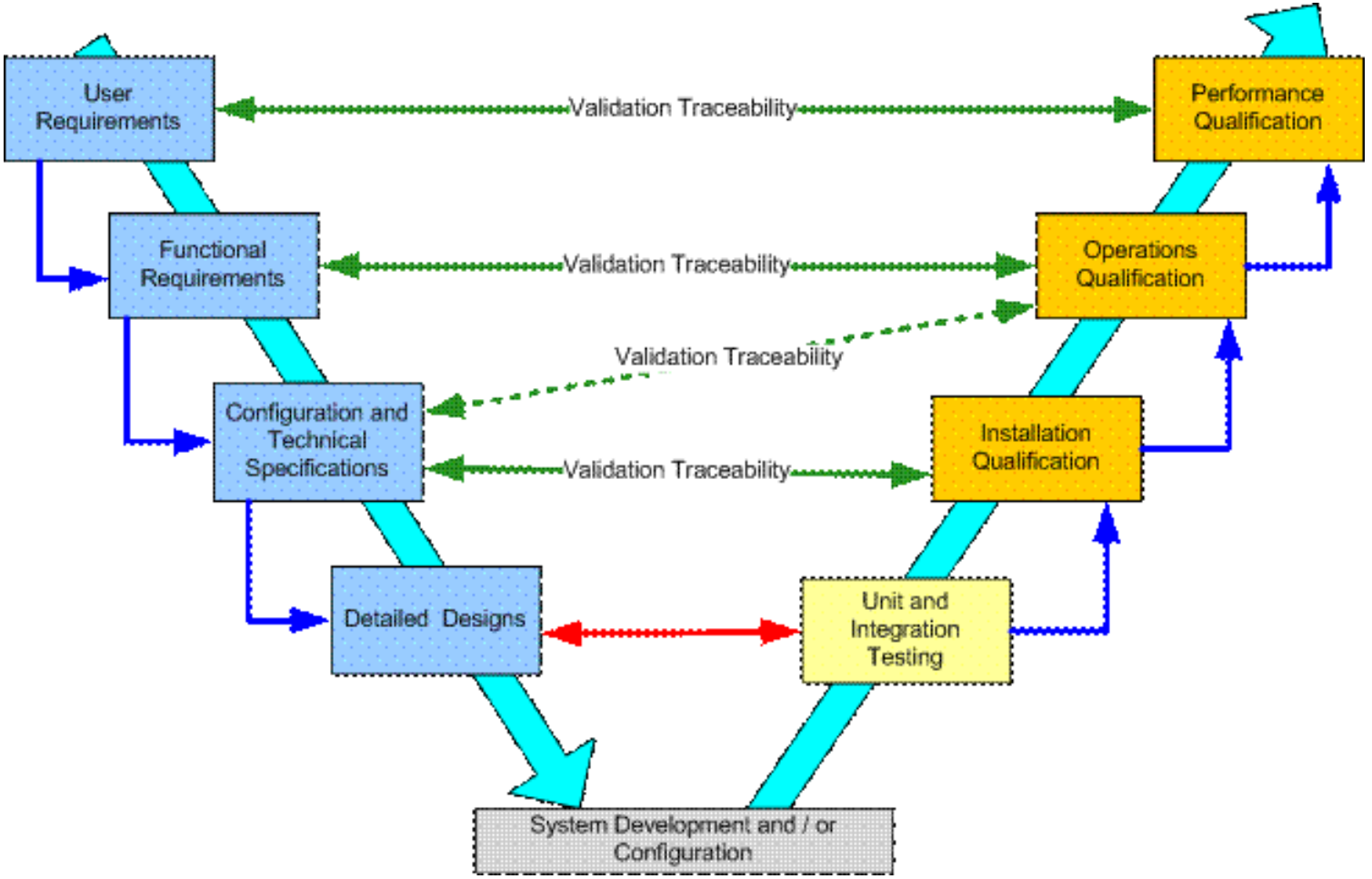


Quelle: Balzert

Das Wasserfallmodell

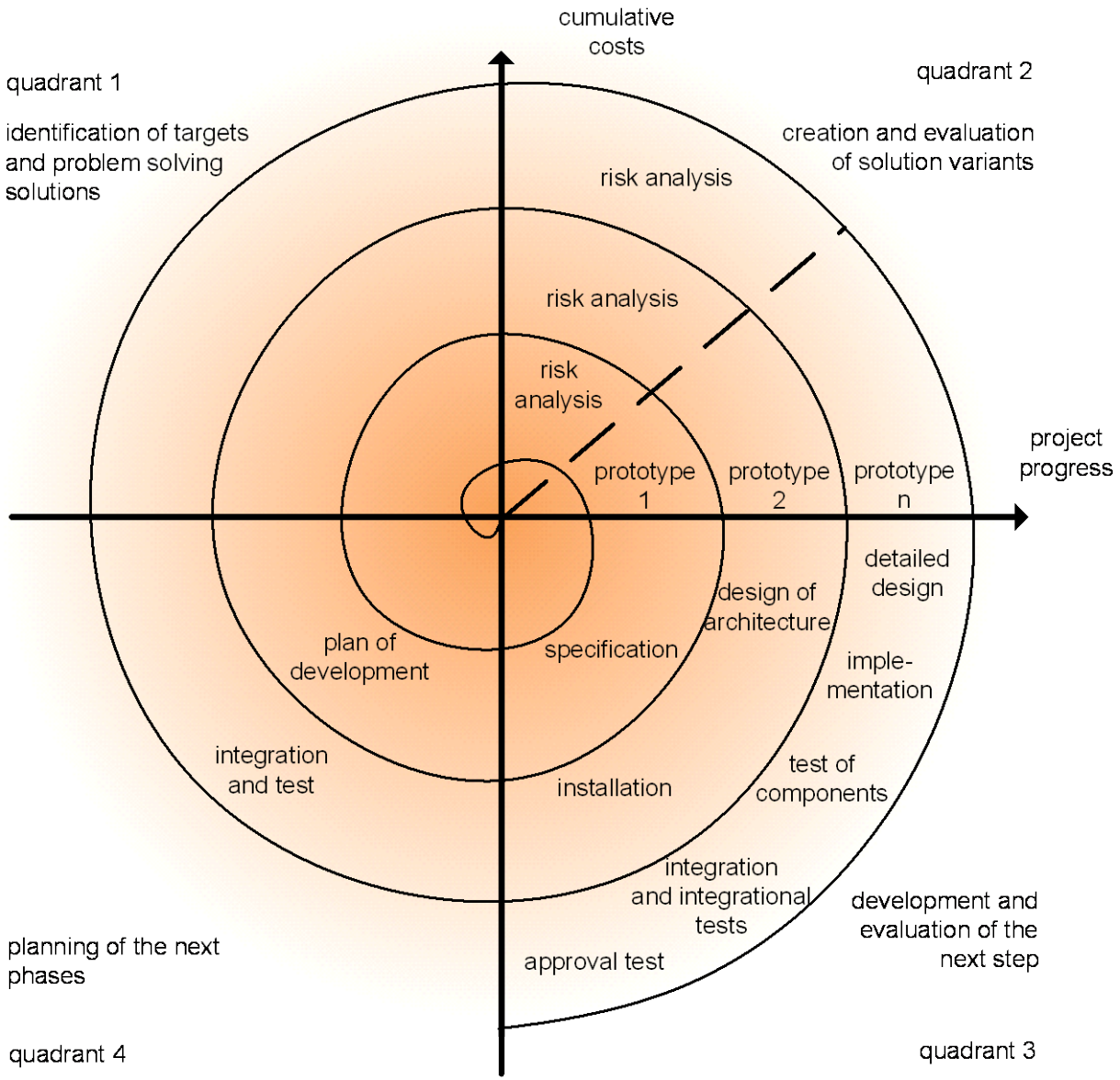


Das V-Modell

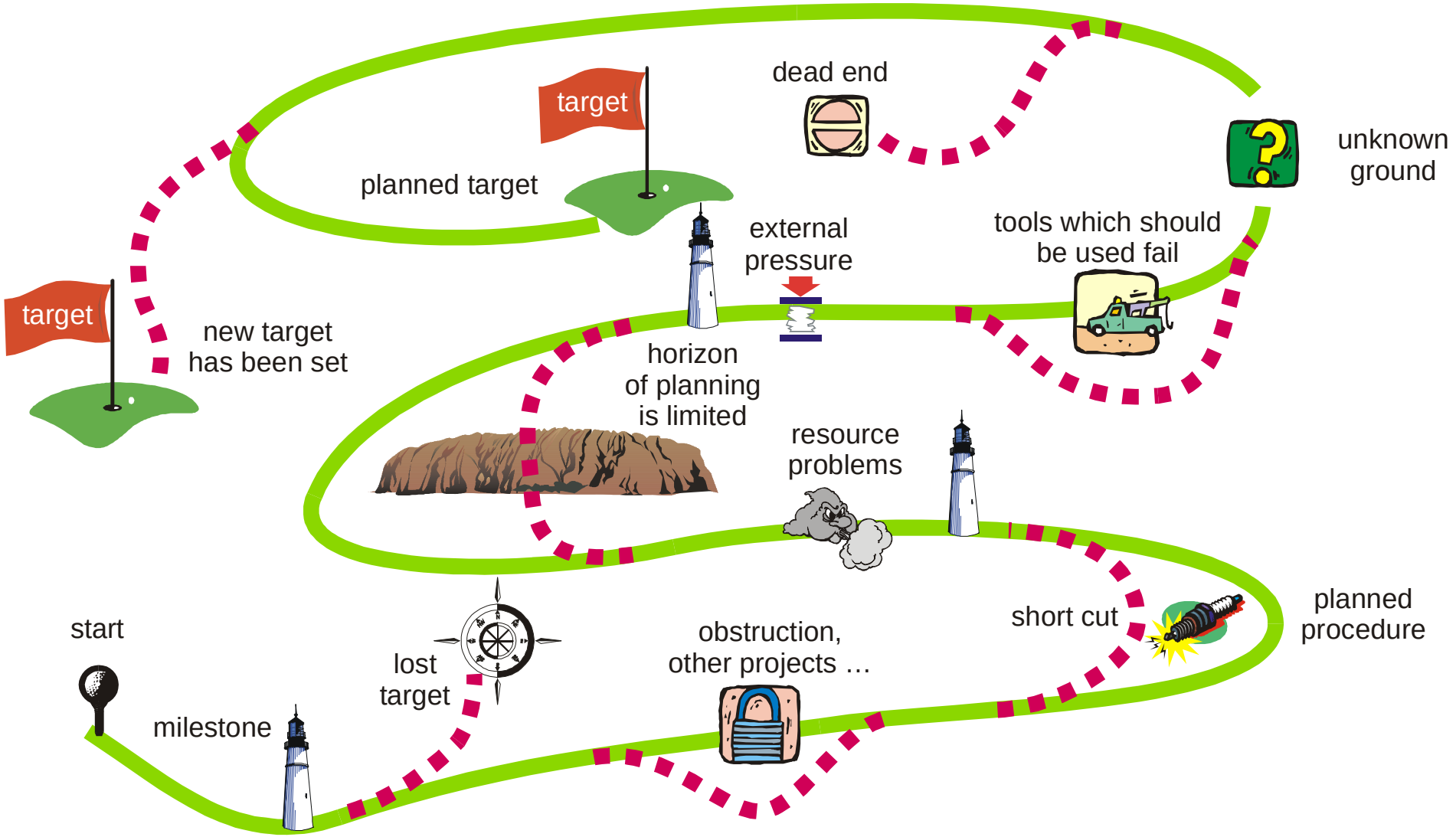


Quelle: Wikipedia

Das Spiralmodell

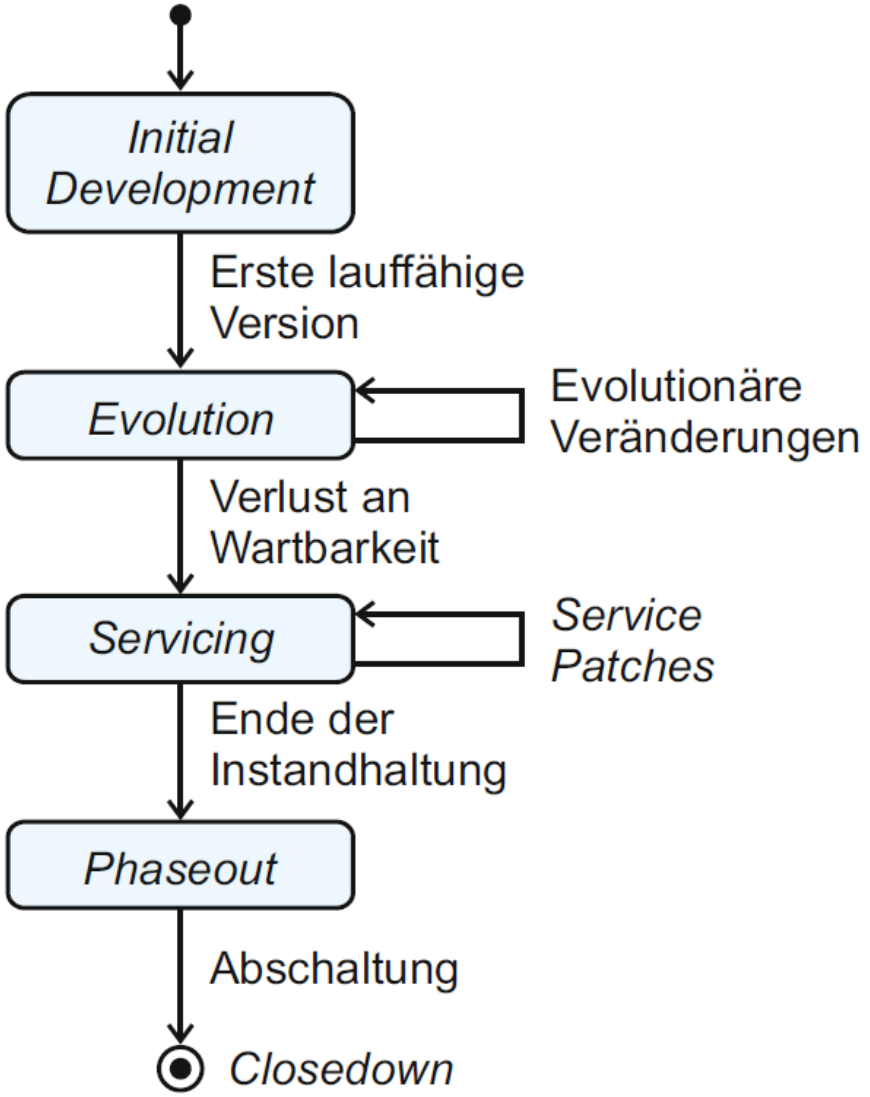


... aber die Wirklichkeit sieht häufig so aus ...



Ein Softwareleben

- 80% des Aufwandes innerhalb eines Softwarelebens entstehen durch Wartung
- Davon sind 40% erforderlich, um die Software zu verstehen.
- Welche Schlußfolgerungen sind daraus für Individual- und Massensoftware zu ziehen
- Wie kann man diesen Sachverhalt
  - in den Griff bekommen, d.h. den Aufwand generell und insbesondere für das Verstehen zu senken
  - durch welches Bezahlmodell unterstützen



Quelle: Balzert

Maßnahmen zur Verbesserung des Softwareentwicklungsprozesses

- Verwendung von
  - Methoden
  - Werkzeugen
  - Richtlinien und Prinzipien

- Unterstützungs-Ziele
  - Planung (Funktionen, Anforderungen, Werkzeuge, …)
  - CASE-Werkzeuge (Computer Aided Software Engineering)
  - Projektmanagement (Zeit-, Kosten- und Ressourcenmanagement)
  - Zusammenarbeit und parallele Arbeit
  - Dokumentation von Funktionsweise, Strukturen und Architektur
  - Fehlerbehandlung und Bugtracking
  - Automatisierte Erstellung und Verteilung
  - Automatisierter Softwaretest, Testszenarien

Anforderungen an eine Programmablaufbeschreibung

- Zustandssicher
  - ist die Beschreibung zeitunabhängig reproduzierbar?
  - sind die Ergebnisse auch bei häufigerer Ausführung noch identifizierbar?
  - sind Quellen und Ziele der Anweisungen bekannt und zugeordnet?
- Unterbrechbar
  - kann eine Anweisung unterbrochen und ohne Verlust zu einem späteren Zeitpunkt fortgesetzt werden?
  - wie stellen sich die Verknüpfungen zu vorab durchlaufenen Schritten dar?
- Elementar
  - beinhaltet eine Anweisung mehrere Schritte?
  - ist die Anweisung eindeutig?
- Gruppierung in Anweisungslisten
  - eine Anweisungsliste erfüllt eine konkrete Aufgabe
  - sie hat einen Namen
  - Festlegung der Ein- und Ausgabekriterien
    - welche Angaben werden im Original übergeben, welche als Kopie,
    - welche Ausgaben sollen bereitgestellt werden
    - in welcher Form erfolgt die Übergabe

UML

- Graphische Modellierungssprache zur Abbildung verschiedenartiger Systeme in Form von 13 unterschiedlichen Diagrammtypen (UML 2)
- Nach ISO standardisiert, d.h. eindeutige Regeln zur Interpretation der UML-Modelle schützen vor Missverständnissen, nicht aber vor Modellierungsfehlern, es existiert also eine eindeutige Definition von Begrifflichkeiten und ihrer Abbildung
- Im Unterschied zu reinen Programmablaufplänen können mit der UML Modelle für statische Strukturen und dynamische Abläufe erstellt werden, dabei beziehen sich die verschiedenen Diagrammtypen aufeinander und ergänzen sich gegenseitig. Durch Einsatz entsprechender Modellierwerkzeuge wird Redundanz vermieden und ein Modellieransatz vom Groben ins Feine unterstützt
- Die UML ist mittlerweile fast zum Standard im Bereich der Softwareentwicklung geworden, aber auch zur Unternehmensmodellierung oder generell zur Systembeschreibung auch anderer (technischer) Systeme geeignet

Aufgabe zum exakten Formulieren

- Beschreiben Sie den Prozess, der beim Leihen eines PKW abläuft
  - Analysieren Sie den Prozess und finden Sie die grundlegenden Prozeduren
  - Beachten Sie die Sichten aus Kunden- und Verleihersicht
  - Beschreiben Sie jeden einzelnen Prozess mit wenigen Worten, wird die Beschreibung zu lang, unterteilen Sie in weitere Subprozesse
  - Versuchen Sie, möglichst viele Aspekte und Randbedingungen in Betracht zu ziehen, nichts zu vergessen
  - Bringen Sie die Prozesse in eine zeitliche Abfolge
  - Beschreiben Sie die Schnittstellen zwischen den Prozeduren anhand folgender Überlegungen:
    Um diese Prozedur zu starten, müssen folgende Dinge vorhanden sein
    Diese Prozedur stellt nach dem Ablauf folgende Dinge zur Verfügung
  - Tauschen Sie Ihr Modell mit anderen und kennzeichnen Sie die Punkte, die nicht exakt genug beschrieben werden, nicht stimmen oder missverstanden werden können

- Nutzen Sie die UML mit Use-Case-, Klassen- und Aktivitätendiagramm

Von der OOA zu OOD (Objektorientiertes Design)

- Nachdem die Funktionsweise eines Softwaresystems prinzipiell in der OOA beschrieben worden ist, erfolgt die weitere Detaillierung im Design (OOD) bereits vor dem Hintergrund der eingesetzten Sprache und Hilfsmittel
- Unterschieden werden kann zwischen
  - Grobdesign (Softwarearchitektur, Systeme und Subsysteme)
  - Feinentwurf (Design der Systeme und Subsysteme selber)
- Beim Grobdesign gilt es folgende Dinge zu berücksichtigen
  - Lebenszyklus (Wartung und spätere Updates, Erweiterungen)
  - Verteilung und technische Infrastruktur (Zielplattformen, Installation, Komponenten, Netzkommunikation, Zusammenspiel …)
  - UML-Komponenten-, Paket- und Verteilungsdiagramme, z.T. Aktivitäten- und Klassendiagramme
- Im Feinentwurf werden die Komponenten der Software designed
  - Festlegung der Funktionsweise der Einzelkomponenten
  - Insbesondere Verwendung von Klassen- und Aktivitätendiagrammen, Sequenzdiagrammen, aber auch Zustandsdiagrammen

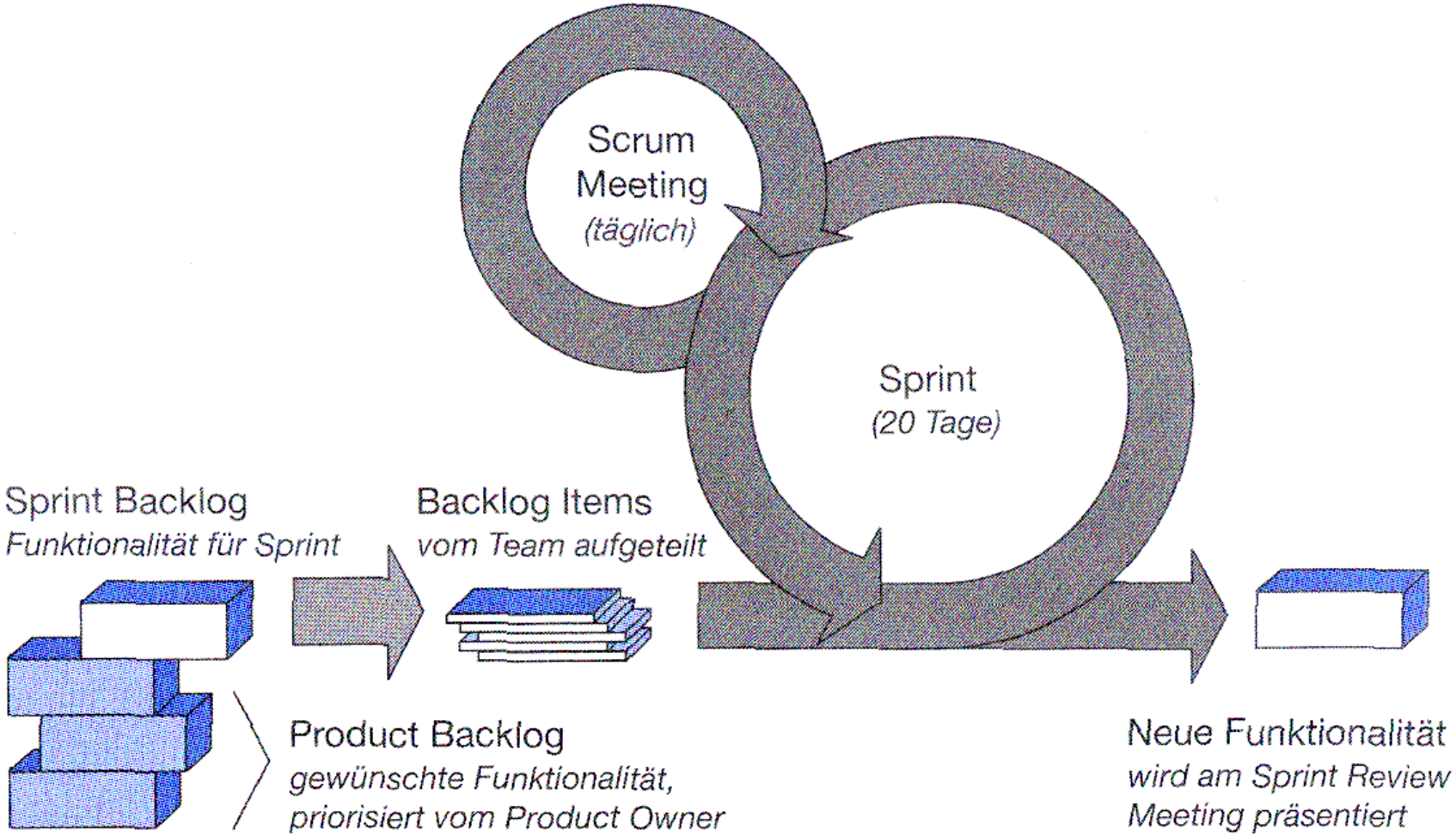
Aufgabe

- Erstellen Sie ein Grobdesign einer Software für einen Autovermieter
  - Nutzen Sie Ihre Überlegungen aus der OOA
  - Berücksichtigen Sie Anforderungen, die sich aus dem tagtäglichen Erleben und Ihrem OOA-Modell ergeben
  - Sehen Sie vor, daß Ihre Software später erweitert werden soll, z.B. um Buchungen direkt aus dem Internet vornehmen zu können.

Agile Softwareentwicklungsprozesse

- Durch das strikte Einhalten eines Softwareentwicklungsprozesses über OOA und OOD kann ein System zwar sehr detailliert vorab beschrieben werden, es ergeben sich aber auch eine Reihe von Nachteilen:
  - Der Kunde sieht „seine Software“ erst sehr spät, wenn Architektur und wesentliche Features schon fast in Stein gemeißelt sind und Änderungen sehr teuer sind
  - Der „Absprung“ von der Planungs- in die Implementierungsphase kann zu früh, aber auch zu spät erfolgen
  - Eine Anpassung des Projektes an sich ändernde Kundenanforderungen ist kaum möglich
  - Das Testen der Software wird tendenziell in späte Phasen verschoben und durch den nahenden Liefertermin auch nicht in erforderlichem Umfang durchgeführt
- Bei den Methoden der agilen Softwareentwicklung wird insbesondere das OOD den Zielen untergeordnet und Mittel zum Zweck. Im Mittelpunkt steht das Softwareprodukt, das inkrementell entwickelt unter Berücksichtigung sich stetig wandelnder Anforderungen wird. Zeitlich begrenzte Entwicklungsphasen wie bei Scrum verhindern ein Verrennen und sorgen für konstanten Projektfortschritt
  - In großen Projekten kommen agile Technologien durch den Koordinationsaufwand an ihre Grenzen und erfordern eine stärkere Einbindung konventioneller OOD-Techniken

Agile Softwareentwicklungsprozesse (hier: Scrum)



Quelle: Grechenik et.al.

Weitere iterative Entwicklungsmethoden

- eXtreme Programming
  - Geeignet für kleine bis mittelgroße Teams
  - Rasche, sich ändernde Anforderungen
  - Hauptziel ist die termingerechte Ablieferung der auftragsgemäßen Software
- Rational Unified Process
  - Definition von Six Best Processes mit
  - Iterative Entwicklung
  - Management der Anforderungen
  - Komponentenbasierte Entwicklung
  - Visuelles Modellieren der Software
  - Prüfen der Softwarequalität
  - Änderungsverfolgung
- …