Oberseminar Softwareentwicklung

Grundlagen der Softwareentwicklung

Inhalt

- Motivation
- Vorschau
- Hintergedanke
- Grundlagen der Softwareentwicklung
- Quellenverzeichnis
- Fragenteil

Inhalt

- Motivation
- Vorschau
- Hintergedanke
- Grundlagen der Softwareentwicklung
- Quellenverzeichnis
- Fragenteil

Motivation

- Softwareentwicklung in großem Maßstab nicht mit „Code – Fix – Strategie“ (blind Reinhacken und Funktion testen) realisierbar
- Verlangen nach Methodik

Inhalt

- Motivation
- Vorschau
- Hintergedanke
- Grundlagen der Softwareentwicklung
- Quellenverzeichnis
- Fragenteil

Vorschau

- Vortrag basiert auf schon vorhandenen Kenntnissen
- leichte Auffrischung des Bestehenden
- keine vertiefende Betrachtung
- Überblick über die !wichtigsten! Aspekte

Inhalt

- Motivation
- Vorschau
- Hintergedanke
- Grundlagen der Softwareentwicklung
- Quellenverzeichnis
- Fragenteil

Hintergedanke I

- Überblick über Stichworte vermitteln; zum selbst Nachschlagen
- Es soll nicht der Eindruck vermittelt werden, dass man alles wissen muss.

Hintergedanke II

- alles Wissen = „Eierlegende Wollmichsau“
  - Unmöglich zu verlangen
    - jedoch nicht verboten es anzustreben
- Im Interesse von jedem selbst zu entscheiden, was wichtig ist und was nicht, wo die Lücken bestehen und wo nicht.

Inhalt

- Motivation
- Vorschau
- Hintergedanke
- Grundlagen der Softwareentwicklung
- Quellenverzeichnis
- Fragenteil

Grundlagen der Softwareentwicklung

- Überblick
  - das Management
  - Technische Grundlagen
  - Grundlagen der Qualitätssicherung

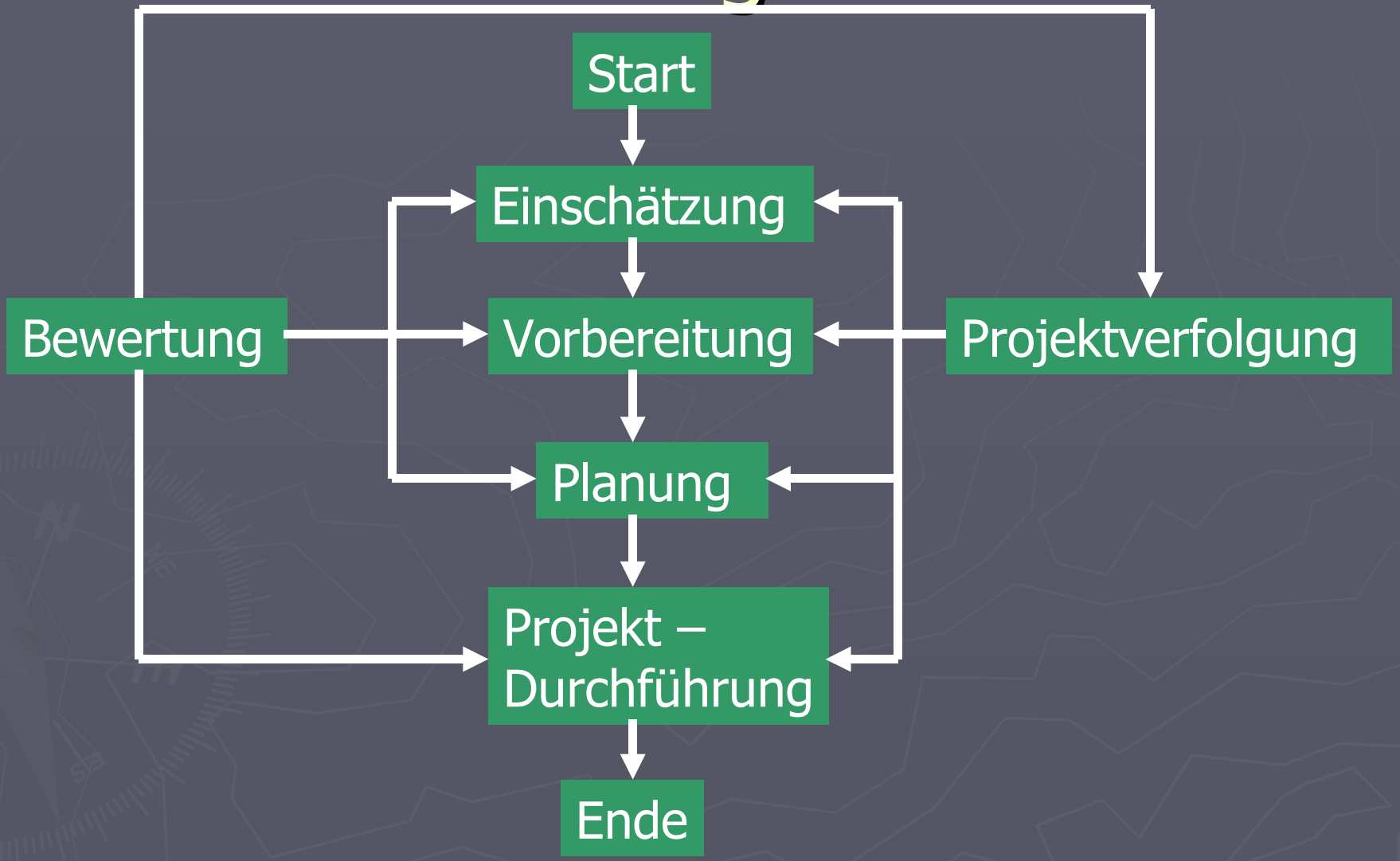
Grundlagen der Softwareentwicklung

- Überblick
  - das Management
  - Technische Grundlagen
  - Grundlagen der Qualitätssicherung

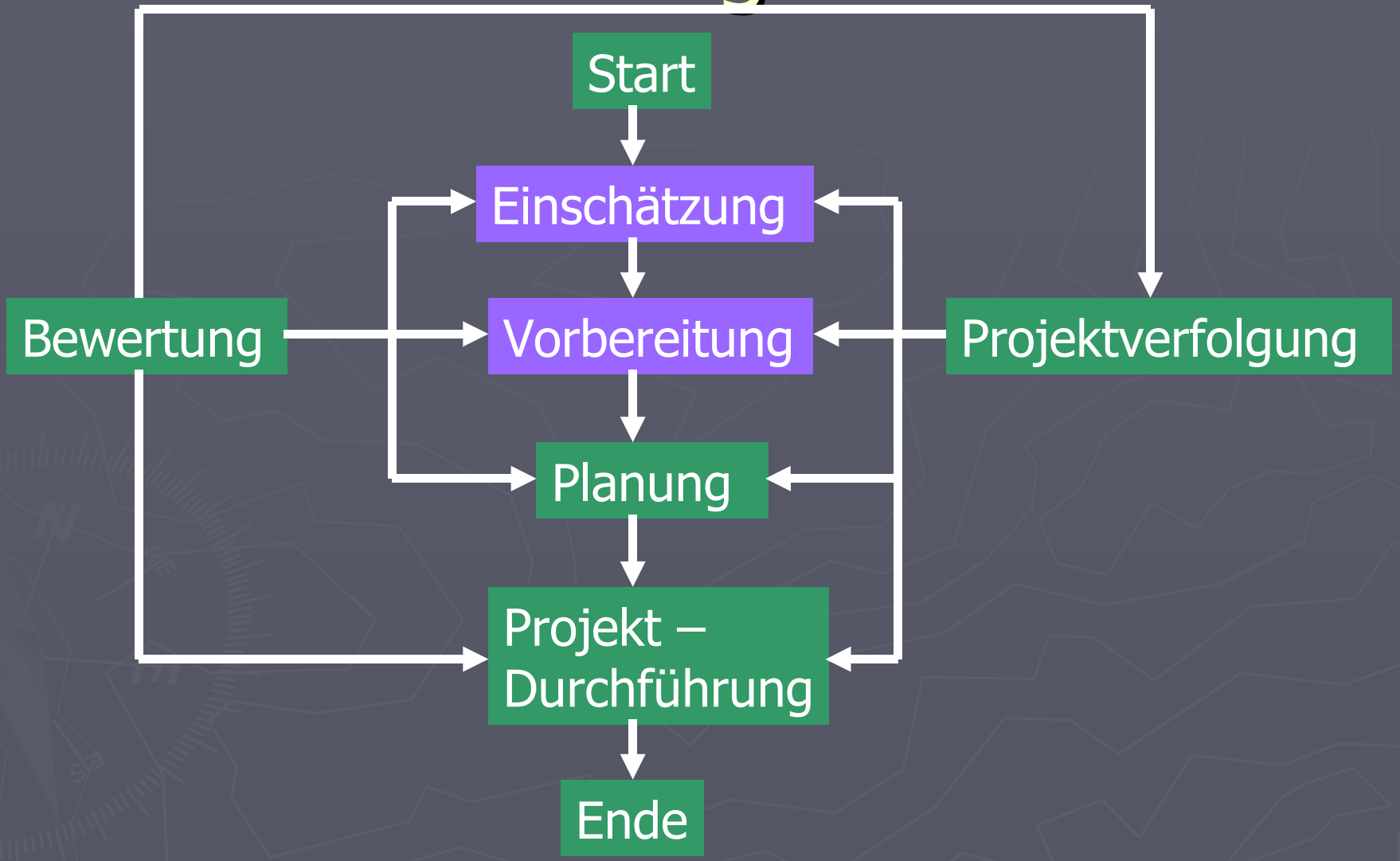
Das Management

- nicht zu unterschätzender Faktor in einem Projekt
- bei kleineren Projekten nicht empfehlenswert (Kosten – Nutzen – Rechnung)
- wichtig zum raschen Erreichen des angestrebten Projektziels

Das Management



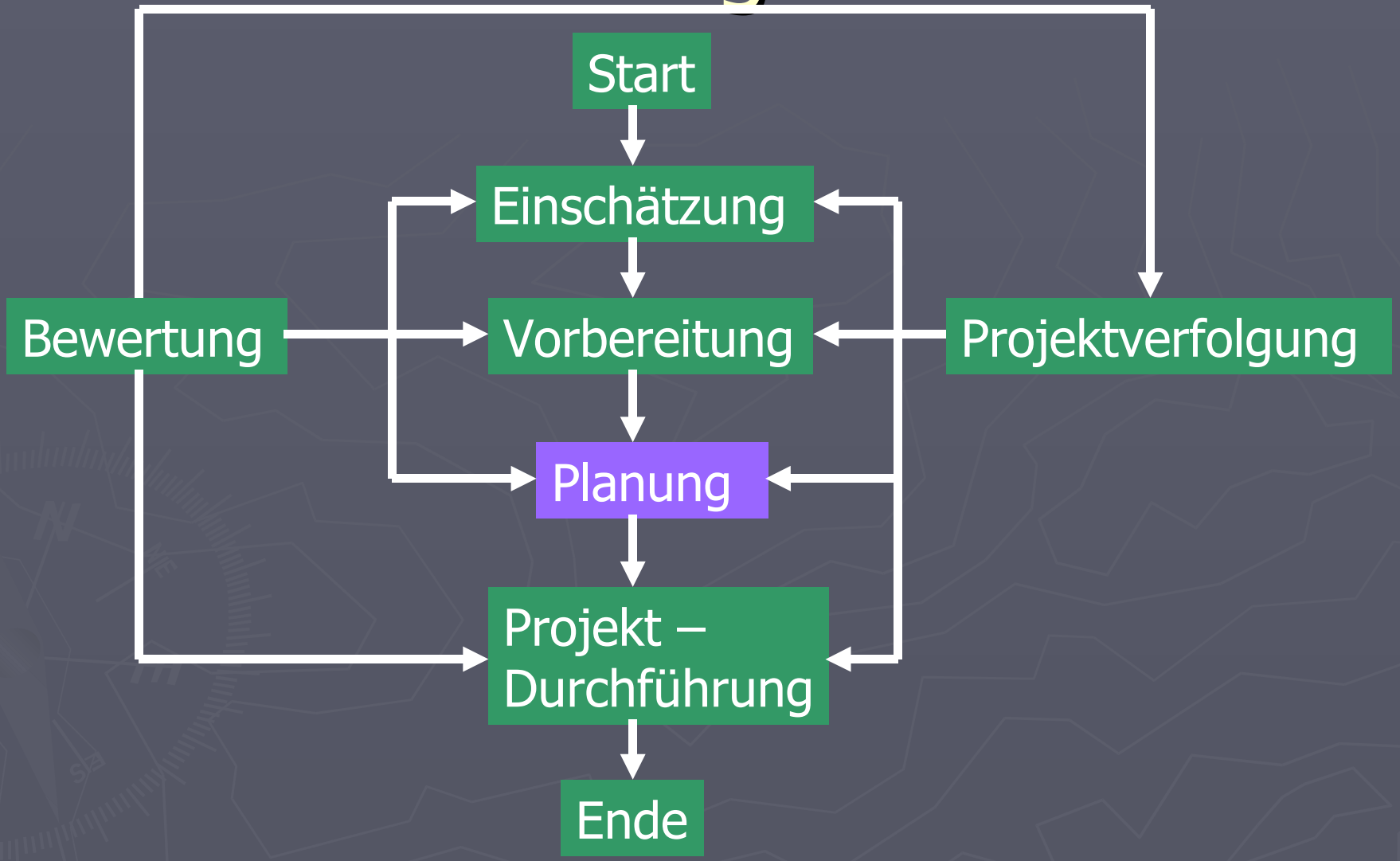
Das Management



Einschätzung und Vorbereitung

- Projektumfang
- Grad des Aufwandes
- daraus Berechnung des Umfangs für die Vorbereitungen (basierend auf Erfahrung und Intuition)

Das Management



Planung

- Auf Basis der Einschätzung Entscheidungen treffen:
  - wie viele Mitarbeiter für Projektrealisierung
  - benötigte Fähigkeiten
  - Planung der Schritte (Aktivitäten)

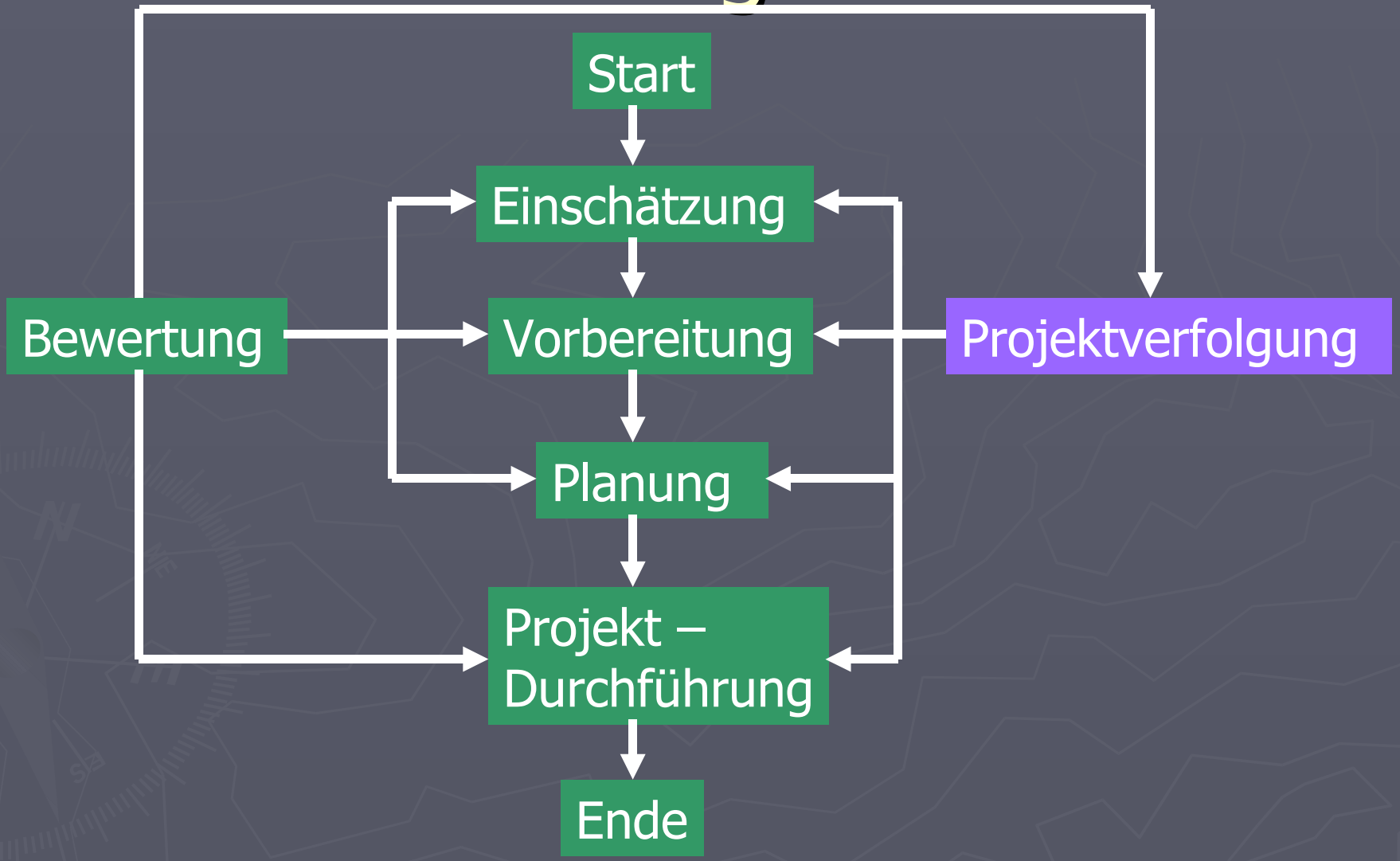
Planung

- Teamzusammensetzung
- „life cycle“ – Model
- zu beachtende Risiken
- Entscheidung, welche Teile der Software selbst implementiert werden müssen und welche Teile bereits fertig genutzt werden können (durch hinzukaufen, Open Source, Lizenzierung)

Planung

- ABER:
  - Vorsicht bei Verpflichtungen
    - nicht überdefinieren (zu restriktives Definieren)
    - nicht zu schwammig (zu viel Freiraum für Projektmitarbeiter, möglicherweise zur Unzufriedenheit des Kunden)
  - kein Druck von Vorgesetzten
  - keine uneinhaltbaren Termine vereinbaren
  - nicht zu viele Änderungen vornehmen

Das Management



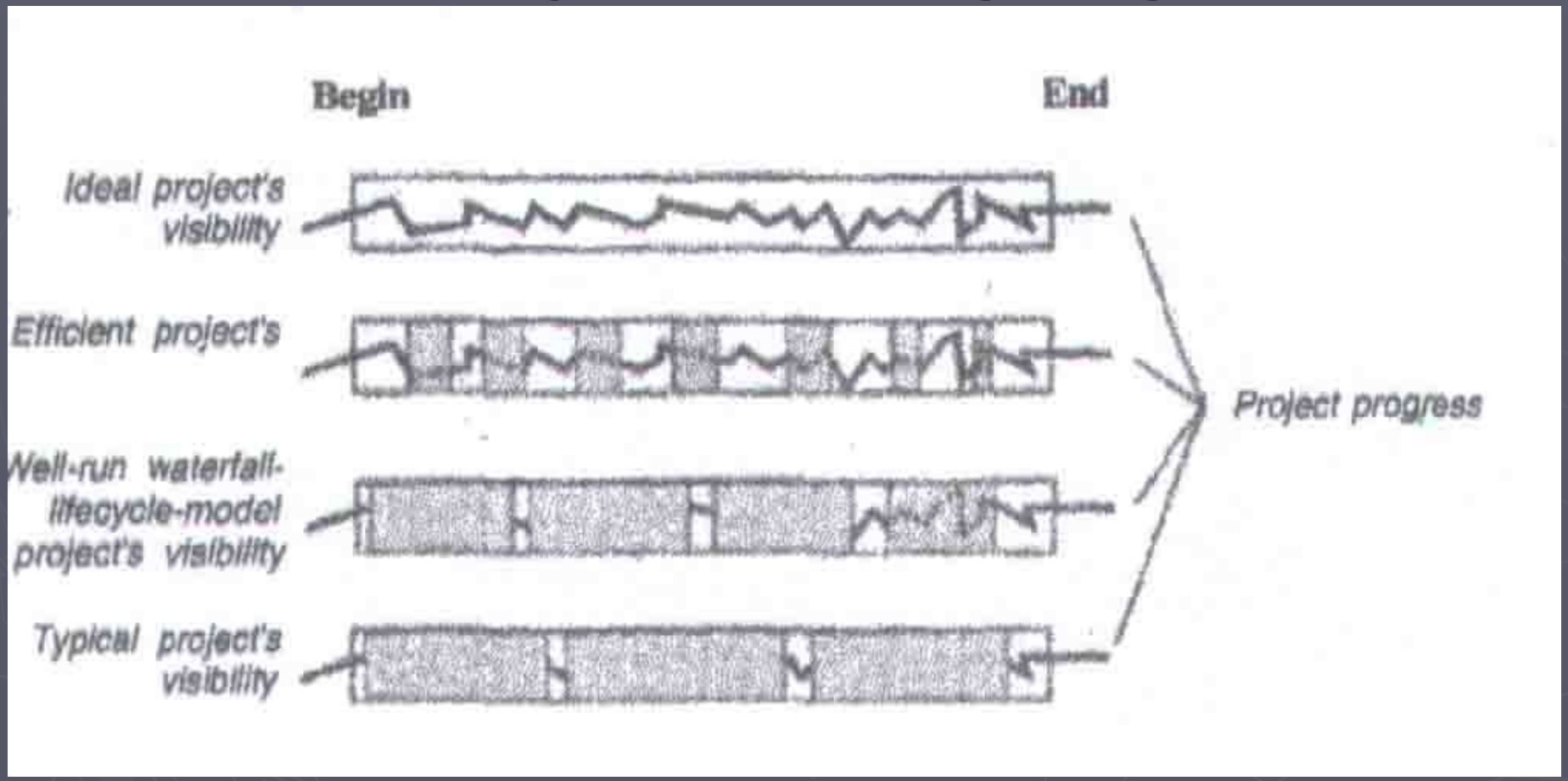
Projektverfolgung

- Projektbegleitende Tätigkeit
- Verifizieren, dass
  - Planung befolgt wird
  - Kosten eingehalten werden
  - Qualität erreicht wird

Projektverfolgung

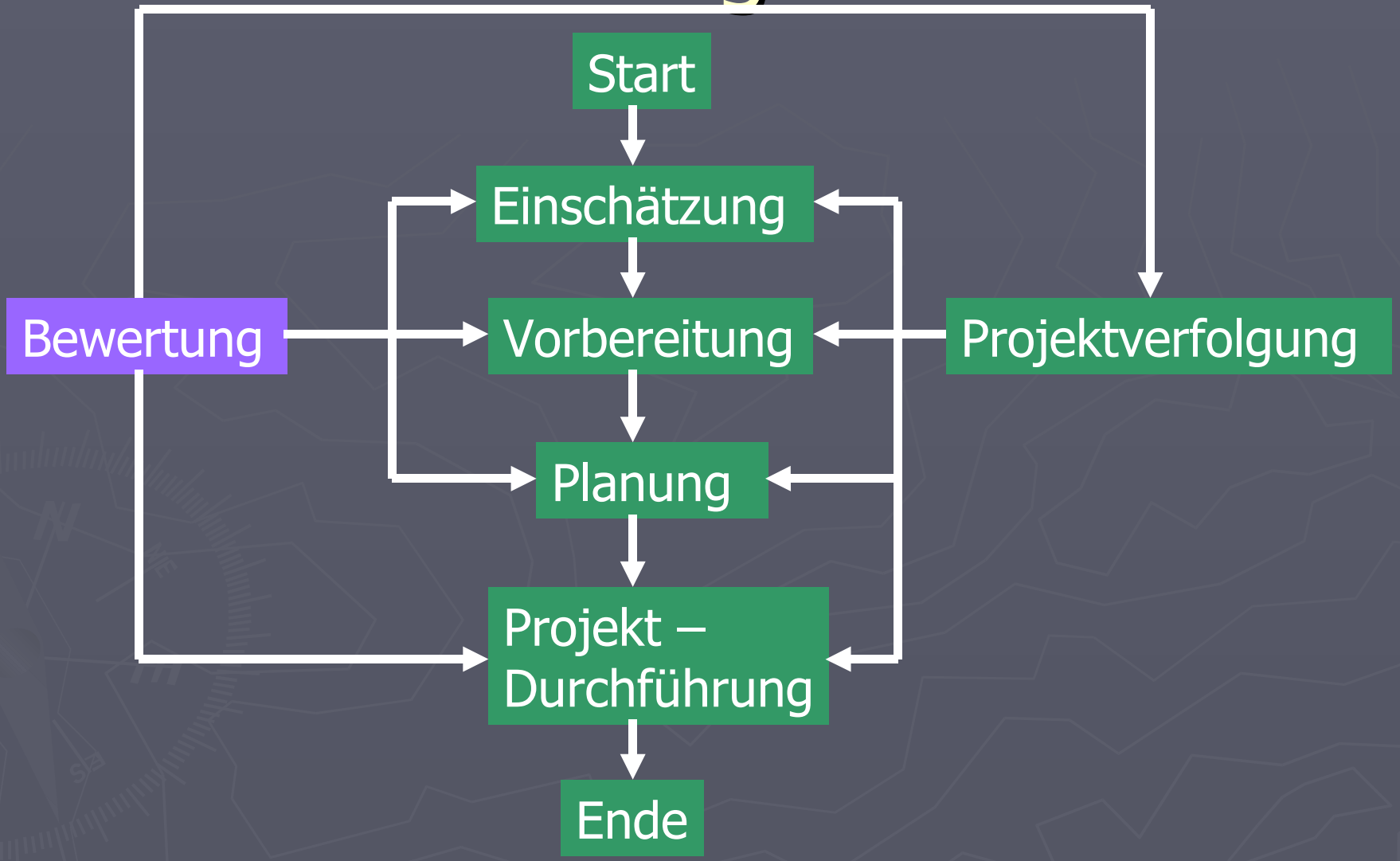
- wichtigstes Instrument überhaupt für Projektmanagement
  - möglich abzeichnende Tendenzen in Projektentwicklung früh erkennbar
  - bei (bevorstehenden) Fehlern Interaktion möglich → Kostensparung

Projektverfolgung



Quelle: „Rapid Development" von Steve McConnell, Seite 57

Das Management



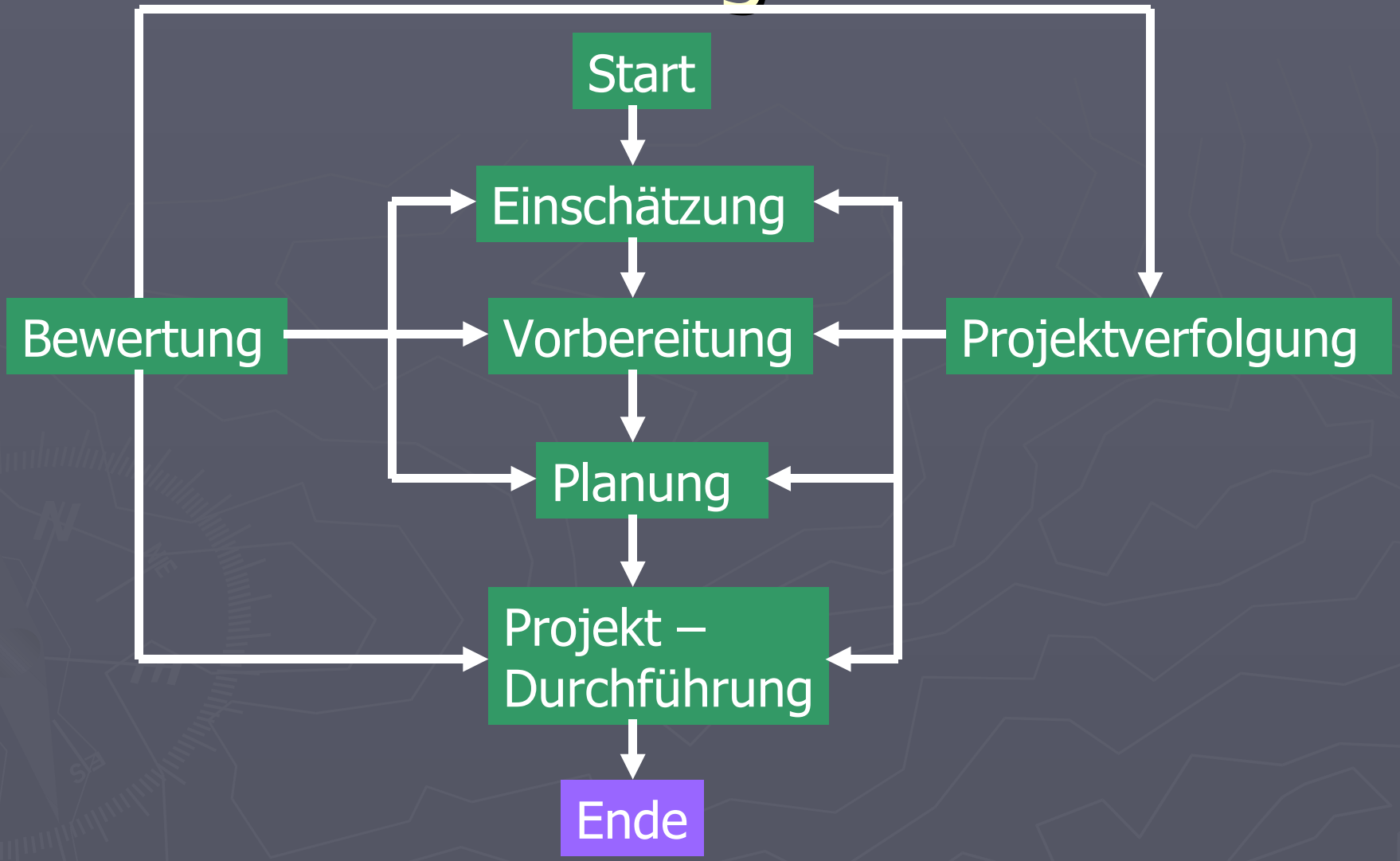
Bewertung

- Sammeln von wichtigen Eckdaten während des Projekts und im Anschluss daran
- z. Bsp.:
  - Kosten- und Zeitplanverläufe
  - Projekthistorie im Bezug auf Mächtigkeit der implementierten Komponenten
  - aufgewendete Arbeitszeit je Projektetappe etc.
  - usw.

Bewertung

- Hintergedanke
  - Projekt bewerten → Erfahrungen sammeln
  - Erfahrungen in spätere Projekte einfließen lassen

Das Management



Grundlagen der Softwareentwicklung

- Überblick
  - das Management
  - Technische Grundlagen
  - Grundlagen der Qualitätssicherung

Technische Grundlagen

- Überblick
  - Moderne Programmierpraktiken
  - Anforderungsmanagement
  - Design
  - Konstruktion
  - Softwarekonfiguration

Technische Grundlagen

- Überblick
  - Moderne Programmierpraktiken
  - Anforderungsmanagement
  - Design
  - Konstruktion
  - Softwarekonfiguration

Anforderungsmanagement

- Kunden- / Entwicklerwünsche werden bereits im Vorfeld des Projekts bzw. während dessen Verlaufs gesammelt
  - In Datenbanken
  - Emails
  - Dokumente
  - usw.

Anforderungsmanagement

- Funktionen werden im Vorfeld bekannt
- erspart das Testen unvollendeter Software durch den Kunden, welcher vorhersehbare Funktionen nachimplementiert wünscht
- bessere Überblick über den Projektumfang
- Verfeinert das Design erheblich

Anforderungsmanagement

- Probleme
  - häufig: Wünsche zu schwammig
    - zu großer Handlungsspielraum für Entwickler
  - seltener: Wünsche zu strikt
    - Wunsch kann bedingt durch das Design nicht so ohne weiteres realisiert werden → große Umwege nötig um Wunsch zu realisieren (teuer + zeitaufwändig)

Anforderungsmanagement

- Problembegegnung
  - UML
  - Designreviews
  - Schnittstelle zwischen Kunden und Entwicklern (Produktmanagement)
  - Sitzungen zwischen Kunden und Entwicklern
  - Ausliefern von Prototypen
  - usw.
  - Verweis auf „Rapid Development“, Seite 62

Technische Grundlagen

- Überblick
  - Moderne Programmierpraktiken
  - Anforderungsmanagement
  - Design
  - Konstruktion
  - Softwarekonfiguration

Design

- Grundbegriffe (Grundkonzepte)
  - z. Bsp.:
    - Objektorientierung
    - Geheimnisprinzip
    - Modularität
    - Vererbung
    - usw.
- Entwurfsmuster

Design

- fortgeschrittene Konzepte
  - Error – Handling (try – catch)
  - Portierbarkeit
  - Datenformate
  - Speicherverwaltung
  - Persistenz der Daten
  - Datenbankdesign
  - Performance
  - Wiederverwendbarkeit bestimmter Komponenten

Design

- spezielle Designfragen
  - branchenspezifische Designentscheidungen
- Architekturfragen
  - 3 – Tier, 4 – Tier

Technische Grundlagen

- Überblick
  - Moderne Programmierpraktiken
  - Anforderungsmanagement
  - Design
  - Konstruktion
  - Softwarekonfiguration

Konstruktion

- Schlagworte
  - Datenbezogene Konzepte (Gültigkeitsbereiche, Speicherung, Wirkungsdauer)
  - Arten von Variablen
  - Coderichtlinien
  - Gliederung der Klassen und Funktionen

Konstruktion

- Teststrategien
- Integrationsstrategien
- Codetuning
- Entwicklertools

- Dennoch beachte Regel:
  - Nicht die Programmiersprache ist entscheidend, sondern ein ausgefeiltes Design.
  - 70% Design – 30% Implementierung

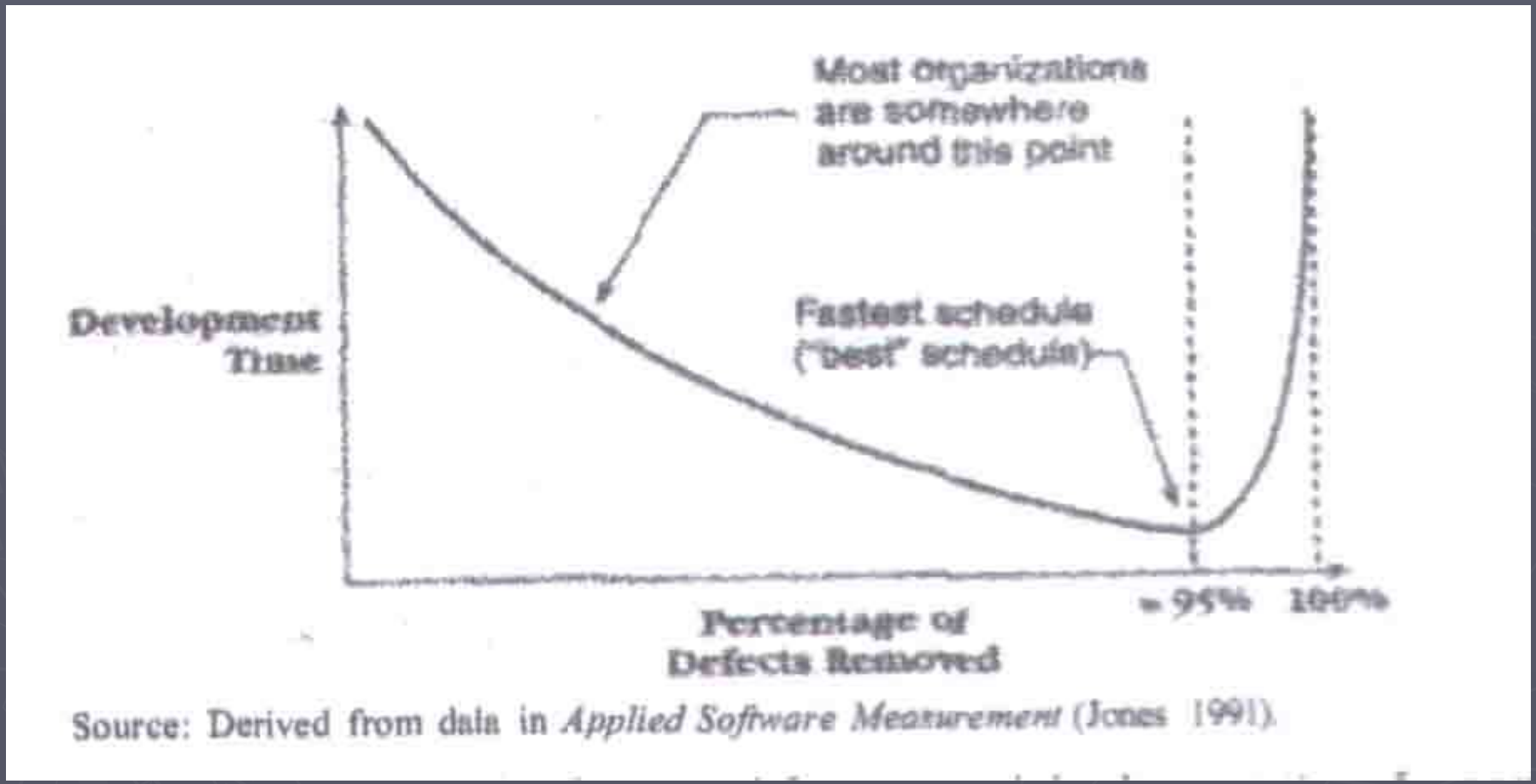
Grundlagen der Softwareentwicklung

- Überblick
  - das Management
  - Technische Grundlagen
  - Grundlagen der Qualitätssicherung

Grundlagen der Qualitätssicherung

- Fehlerhafte Module
- Testverfahren
- Technische Begutachtung
  - Durchdenken
  - Codeverifikation
  - Inspektionen

Grundlagen der Qualitätssicherung



Source: Derived from data in *Applied Software Measurement* (Jones 1991).

Quelle: „Rapid Development" von Steve McConnell, Seite 69

Testverfahren

- Unit - Tests und Systemtest
- Unit – Tests finden 10 – 50% der Defekte in einem Programm (Entwickler testet den eignen Code)
- Systemtests finden 20 – 60% dieser Fehler
- Kombinationen aus beiden oft weniger als 60%

Technische Begutachtung

- Anwenden auf Implementierung, Design, Anwendungsfälle, Testszenarien usw.
- zeitkritische Komponente
- Im Vorfeld vor der Implementierung anwendbar
  - günstiger als Testen allein

Technische Begutachtung

- Durchdenken
  - meist informell, „Entwickler sitzen im Kreis“ und überarbeiten bereits bestehende Funktionen/Spezifikationen etc. mit dem Ziel ihrer Verbesserung
  - finden in der Regel 30 – 70 Prozent aller Fehler in einem Programm/Projekt

Technische Begutachtung

- Codeverifikation
  - Entwickler überfliegen Code und melden Fehler an dessen Autor
  - findet zweimal mehr Fehler pro Stunde als Testen

Technische Begutachtung

- Inspektion
  - beste Methode zur Qualitätssicherung
  - Zur Erinnerung (grob):
    - „Moderator" händigt zu verifizierendes Produkt an die „Reviewer" aus
    - „Reviewer" inspizieren Produkt anhand von Checklisten
    - Urheber des Produktes kommentiert die inspizierten Stellen (was soll damit getan werden), die „Reviewer" schreiben dazu Einträge, welche auf getretene Fehler genau beschreiben

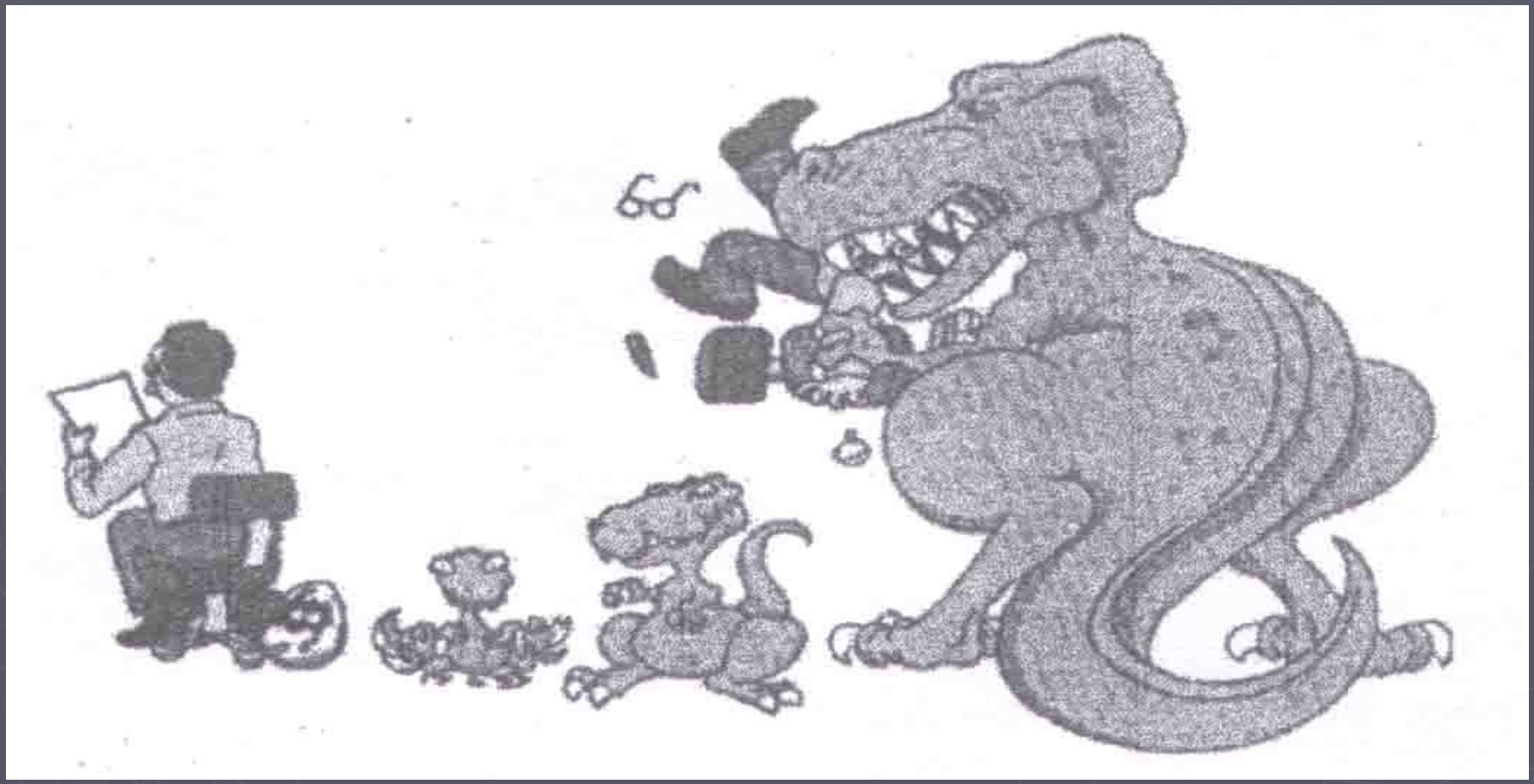
Technische Begutachtung

- „Moderator“ verfasst anschließend eine Bericht zur Inspektion mit allen gefundenen Fehlern mit Hinweis auf die Lösungsmöglichkeiten
- findet 60 – 90% der Fehler in einem Programm
- nach Steve McConnell: „1 Stunde die in eine Inspektion gesteckt wird, erspart bis zu 33 Stunden aufwendiger Wartungsarbeiten.“ (sinngemäß)

Grundlagen der Qualitätssicherung

- Ranking der Methoden zur Qualitätssicherung nach Steve McConnell
  - technische Begutachtung besser als normales Testen
  - Testen findet nur Symptome ←→ technische Begutachtung finden die Ursachen des Defekts und dessen Symptome

Grundlagen der Qualitätssicherung



Quelle: „Rapid Development" von Steve McConnell, Seite 75

Quellenverzeichnis

- Steve McConnell „Rapid Development“
- www.google.de