




1 Einführung in Software Engineering

1.1 Motivation: Warum ist SE wichtig

1.1.1 Terminologie

Definition: **Software Engineering (SE)** [nach Balzert 2009 synonym für Softwaretechnik]

Zielorientierte Bereitstellung und systematische Verwendung von **Prinzipien, Methoden, Konzepten, Notationen** und **Werkzeugen** für die

- **arbeitsteilige,**  **Software-Projekt**
- **ingenieurmäßige**  **Entwicklungsprozess**
Entwicklung und Anwendung von
- **umfangreichen**  **Industrielle Problemstellung**
Software-Systemen.

- **Zielorientiert** bedeutet die Berücksichtigung u.a. von **Kosten, Zeit, Qualität**
- Viele Projekte scheitern daran das die kein vernünftiges SE haben !!

1.1.2 Death March Projects

Definition: Wahrscheinlichkeit mit dem Projekt scheitert ist > 50%

Projekteigenschaften

- Projektlaufzeit wurde auf Hälfte der ursprünglichen Planung reduziert
- Projektteam wurde auf Hälfte reduziert
- Budget und Ressourcen wurden auf Hälfte reduziert
- Funktionalität, Eigenschaften, Performance- und sonstige technische Anforderungen sind doppelt so groß wie unter normalen Umständen

Warum passieren Death March Projects?

politische Gründe
("politics, politics, politics")

- Projekt soll unbedingt durchgeführt werden
- Projektbeteiligte agieren gegeneinander

Naivität/ Unerfahrenheit

- 3-Jahresprojekt wird auf 9 Monate geschätzt
- unterschätzen der Fachlichkeit

"hysterischer Optimismus"
(Tom DeMarco)

- „Problem XYZ programmieren wir an einem Wochenende“

unprofessionelle IT-Firmen
(u.a. Start-Ups)

- oft unterfinanziert, kein ausreichendes Management, optimistisch, unzureichend ausgebildetes oder wenig Personal

Marine Corp Mentalität

- „richtige Programmierer*innen schlafen nicht“: Überstunden und höchste Belastung der Entwickler*innen als Erfolgsrezept

hoher Wettbewerbsdruck

- Globalisierung – Liberalisierung (z.B. Telekommunikations-/ Energiemarkt)
- **Festpreisprojekte**

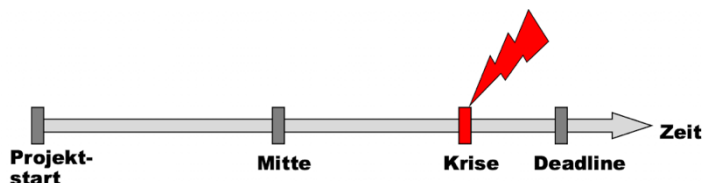
neue Technologien

- Verwendung noch nicht ausgereifter Technologie, ohne Erfahrungen

unerwartete Krisen

- Entwickler*innen kündigen, Produkte sind nicht verfügbar,...

- typischer Projektverlauf von Death March Projekten



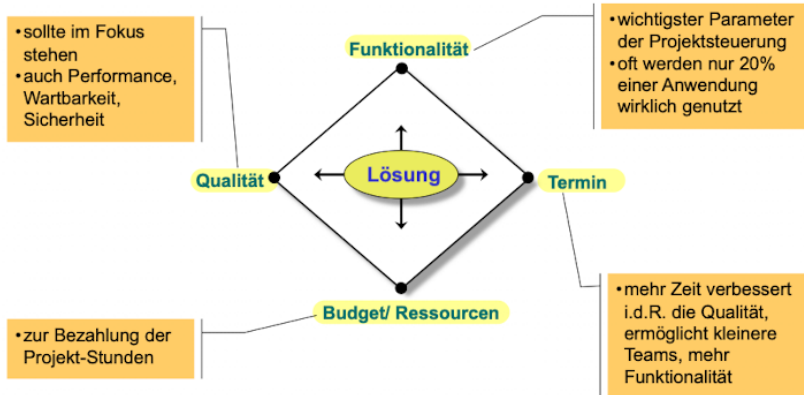
- erst (zu) spät wird deutlich, dass Projektplanung nicht haltbar ist

→ **Steuerungsmechanismen** (= Erfolgsfaktoren) für Projekte erforderlich, um solche **Krisensituationen** frühzeitig zu erkennen und zu vermeiden

1.2 Erfolgsfaktoren für Projekte

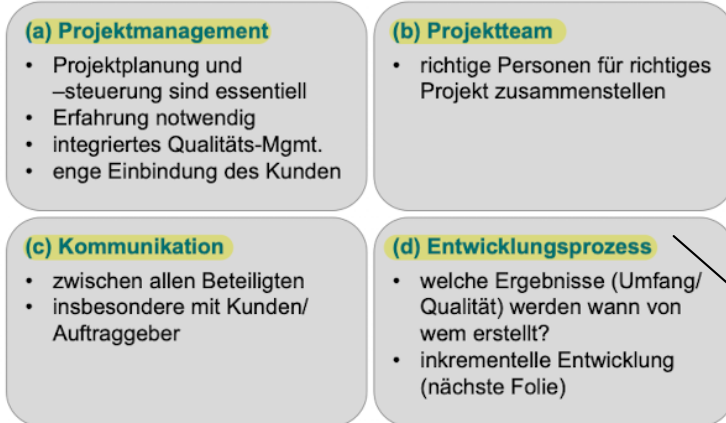
1.2.1 Einflussgrößen

- **Erfolgreiches Projekt** = Wechselwirkung zwischen Einflussgrößen ausbalancieren



1.2.2 Erfolgsfaktoren (= Gegenstand des Software Engineering)

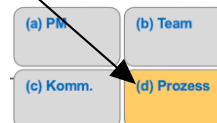
- durch folgende Steuerungsmechanismen lassen sich auch schwierige SW-Entwicklungsprojekte erfolgreich durchführen



Unified Process

(d) Entwicklungsprozess: Inkrementelle Entwicklung

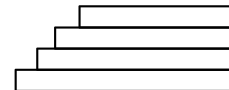
- komplexe Software sollte **stufenweise erstellt** werden (Software-Versionen werden von Stufe zu Stufe erweitert)
- Projekt über Funktionsumfang steuern
- Priorisieren der Anforderungen
 - zuerst das realisieren, was dem Benutzer den größten Nutzen bringt
 - oft werden nur 20% einer Anwendung wirklich genutzt !



- **Risiken frühzeitig** angehen, d.h. wichtigste Anforderungen zuerst

⇒ **inkrementeller Entwicklungsprozess** notwendig bzw. anzustreben

⇒ Unified Process und agile Methoden (z.B. Scrum) sind inkrementell

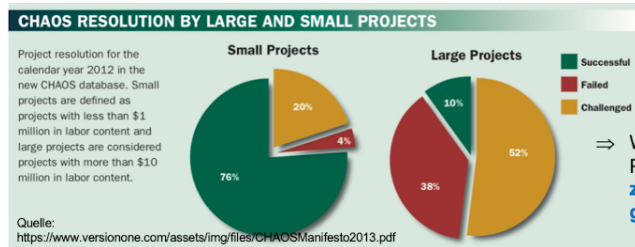


1.3 Warum ist Software-Entwicklung so schwierig?

1.3.1 Gründe für Probleme bei Software-Entwicklung

- Software products are among the **most complex of man-made systems**, and software by its very nature has intrinsic, essential properties (e.g., **complexity, invisibility, and changeability**) that are not easily addressed
- Programming techniques and processes that worked effectively for a small team to develop modest-sized programs **do not scale-up well** to the development of **large, complex systems** (i.e., systems with millions of lines of code, requiring years of work, by hundreds of software developers)
- The **pace of change in** computer and software **technology** drives the demand for new and evolved software products. This situation has created customer expectations and competitive forces that strain our ability to produce quality of software within acceptable development schedules.

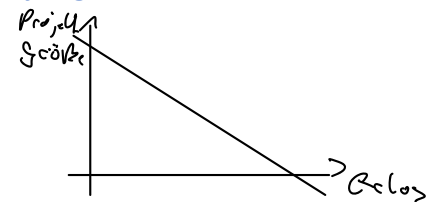
1.3.2 Projektrisiken, Projektgröße, Komplexität



→ Wahrscheinlichkeit eines Projekterfolgs **nimmt mit zunehmender Projektgröße** dramatisch ab



→ **Komplexität** und **Größe** (beteiligte / Personen, Dauer etc.) von SW- a Projekten steigen **kontinuierlich**



Zunehmende Komplexität

Year	Systems	SLOC (Million)
1993	Windows NT 3.1	4-5
2000	Windows 2000	>29
2001	Windows XP	40
2015	Windows 10	50
2007	SAP NetWeaver	238
2017	Google Services	2.000

1.3.3 Was ist anders bei industrieller SW-Entwicklung als bei "SW-Entwicklung im Kleinen" oder akademischer SW-Entwicklung?

- **industrieller Maßstab**
 - Bearbeiterjahre, Quellcodezeilen im Bereich Hunderttausend bis zu Millionen
- **Komplexität durch Größe, Technik und fachliche Vielfalt**
 - Komplexität eher durch **Umfang und Breite**, weniger durch knifflige Algorithmik
- **Verhältnis Netto- zu Brutto-Code**
 - **Netto-Code** löst eigentliches Problem, **Brutto-Code** zusätzl. Zur Fehlerbehandlung, Hilfsfunktionen, Berechtigungsprüfung etc.
 - universitäre SW Verhältnis ca. 1, industrielle SW 0,5 bis 0,25
- **Qualitätsanforderungen** (z.B. Fehlerfreiheit, Performanz etc.)
- **Großes Datenvolumen**
 - Transaktionen, Batch-Prozesse etc.
- **Integration in existierende Systemwelt**
 - Migration, Parallelbetrieb, Heterogenität etc.
- **Kunde, Kunde, Kunde**

1.4 Zusammenfassung

- industrielle Software-Entwicklung hat **enorme Fortschritte** erzielt, ist aber immer noch nicht **vollständig ingenieurmäßig durchdrungen**
- Voraussetzungen für erfolgreiches Projekt
 - **Entwicklungsprozess** (= systematisches Vorgehen) (⇒ SE1)
 - **Software Design** (= Entwurf qualitativ hochwertiger Software)
 - **Projekt-/ Qualitätsmanagement** zur Planung, Organisation, Steuerung, Kontrolle von Projekten (⇒ SE2 und Master-Studiengang)
- ⇒ **Software Engineering**
- Software Engineering: der Bereich in Informatik, der sich mit ingenieurmäßiger Entwicklung von Software befasst
- Softwaretechnik wird häufig synonym zu Software Engineering verwendet