„Classic Mistakes“

Klassische Fehler in der Softwareentwicklung

Jan Hoffmeyer

01-IND

Übersicht

- Einführung
- „Klassische“ Fehler
- Übersicht
- Gilligan‘s Insel
- Der Ausweg
- Quellen

Einführung

- Kompliziert
- Genug Gelegenheiten
- „One bad apple can spoil your whole project“
- Moderne Methoden
- Ursache
- Schnell <> Langsam (Entwicklung)

„Klassische“ Fehler

- Der Mensch
- Der Ablauf
- Das Produkt
- Die Technologie

„Klassische“ Fehler – Mensch
Untergrabene Motivation

- → Produktivität und Qualität
- Mangelnde Konservation
- Urlaub des Manager
- Bonus zum Abschluss

„Klassische“ Fehler – Mensch
Hinzufügen von Personen

- Klassischster Fehler
- Trugschluss
- Gegenteil

„Klassische“ Fehler – Mensch
Spannung zwischen Entwickler und Kunde

- Viele Ursachen z.B. Zeitplan
- Konflikte
- Wenige Kommunikation → Anforderung, UI, komplette Produkt
- Zeit verbrauchend und ablenkend

„Klassische“ Fehler – Mensch
Unrealistische Erwartungen

- Spannung Entwickler und Kunde
- Unrealistische Hoffnungen
- Verlängern nicht, aber tragen zur Auffassung bei

„Klassische“ Fehler – Mensch
Mangelnde Benutzereinbeziehung

- Benutzer zuständig für Erfolg
- Kein Bezug zum Benutzer → Risiko von Missverständnissen und Verwundbarkeit

„Klassische“ Fehler – Mensch
Politik über Inhalt

- Vier politische Orientierungen
  - Politisch
  - Forschend
  - Isolierend
  - Allgemein
- Politisch fatal für schnelle Entwicklung

„Klassische“ Fehler – Mensch
Wunschdenken

- Nicht nur Optimismus
- Großer Knall
- Untergräbt sinnvolle Planung
- Wurzel vieler Softwareprobleme

„Klassische“ Fehler – Ablauf
Zu optimistische Terminpläne

- Unterschiedliche Terminpläne
- → Projektrahmen wird nicht erfüllt
- → keine effektives Planen
- → Verkürzen wichtiger Schritte (Analyse, Design)
- Druck auf Entwickler → Moral und Produktivität

„Klassische“ Fehler – Ablauf Lieferantenfehler

- Auslagerung von Teilprojekten
- Einfluss auf Zeit, Qualität, Spezifikation
- Verlangsamen statt beschleunigen

„Klassische“ Fehler – Ablauf
Fuzzy Front End

- Zeit der Genehmigung und Finanzierung
- Gefahren
  - Aggressiver Zeitplan
  - OOD

„Klassische“ Fehler – Ablauf
Streichung von Entwicklungsschritten

- Zeitnot → Herausstreichen nicht wichtiger Schritte
- Kein Code produzierend → leichtes Ziel (Anforderungsanalyse, Architektur, Design)
- „jumping into coding“

„Klassische“ Fehler – Ablauf
Unzureichendes Design

- Folgefehler der Streichung
- Wenig Zeit fürs Design (nicht anpassungsfähig)
- Design meistens wichtiger als Qualität

„Klassische“ Fehler – Ablauf
Streichung der Qualitätszusicherung

- Eliminierung von Design und Code Kontrolle
- Keine Testplans, nur Funktionstest
- 1 Tag gespart, 3-10 Tage drangehängt

„Klassische“ Fehler – Ablauf
Überhastete Annäherung

- Kurz vor Ende des Projektes übereilte Fertigstellung
- Z.B. Performance, Dokumentation, Hilfesystem, …
- Häufig zu oft → verschwendet Zeit und verlängert Planung

„Klassische“ Fehler – Ablauf
Planen zum späteren Auffangen

- Unzureichende Reaktion auf Abweichen von Planung
- Produktänderung (Anforderung, Design, …)

„Klassische“ Fehler – Ablauf
Code-like-hell Programmierung

- All-as-you-go Programmierung
- Motivierte Entwickler meistern jedes Hindernis
- Unternehmerannäherung
- Kombination von Code-And-Fix mit ehrgeiziger Planung → funktioniert nicht

„Klassische“ Fehler – Produkt
Goldgepresste Bedürfnisse

- Mehr Bedürfnisse wie gebraucht werden
- Z.B. Performance
- Vernachlässigung wichtiger Merkmale (Marketing, Entwicklung, …)
- Nicht genug berücksichtigt in Planung

„Klassische“ Fehler – Produkt
Goldgepresste Entwickler

- Faszination neuer Techniken
- Übernehmen von interessanten Merkmalen
- Verlängerung der Planung

„Klassische“ Fehler – Produkt Forschungsorientierte Entwicklung

- Seymour Cray: Entwicklung in mehr als 2 Gebieten zur gleichen Zeit birgt eine hohes Risiko von Fehlern
- Erstellung neuer Algorithmen → Softwareforschung (nicht vorhersehbar)
- State of the Art – Algorithmen

„Klassische“ Fehler – Technologie Silberkugel Syndrom

- Glaube: ein Tool allein kann Wunder verbringen
- Zu wenig Information über Nutzen und Wirken
- Einsatz als Problemlösung → zwangsläufige Enttäuschung

„Klassische“ Fehler – Technologie
Überschätztes Einsparung durch neue Praktiken

- Selten gigantische Sprünge der Produktivität
- Erlernen des Umganges neuer Praktiken beansprucht Zeit
- Neue Risiken
- Wiederverwendung von Codesegmenten

„Klassische“ Fehler – Technologie
Toolwechsel während des Projektes

- Alte Praktik
- Versionsupdate manchmal sinnvoll
- Lernkurve, Neuarbeit und unvorhersehbare Fehler → zerstören jeden Vorteil

„Klassische“ Fehler – Technologie
Mangel automatischer SCC

- Überflüssiges Risiko
- Manuelle Koordination
- Gefahren durch Verwendung alter Code – Versionen
- Im Durchschnitt 10% Code – Änderung mit SCC (pro Monat)

„Klassische“ Fehler
Zusammenfassung – Menschliche Fehler

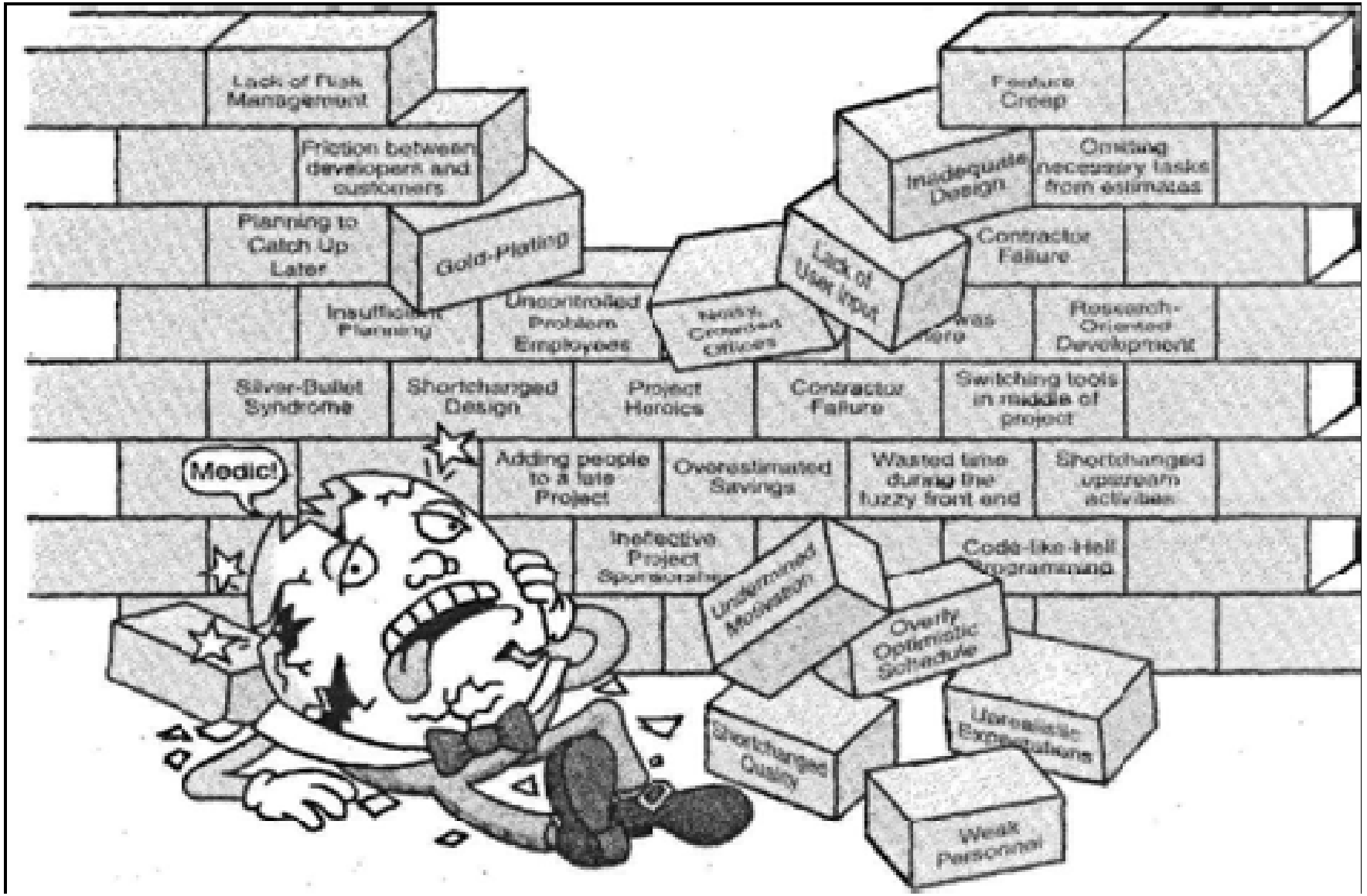
Untergrabene Motivation	Hinzufügen von Personen	Mangelnde Projekt-unterstützung	Wunschdenken
Schwaches Personen	Lautes Büro	Mangelnder Teilhabereinkauf	
Unkontrollierte Problemarbeiter	Spannung zwischen Entwickler und Kunde	Mangelnde Benutzereinbeziehung	
Helden	Unrealistische Erwartungen	Politik über Inhalt	

„Klassische“ Fehler
Zusammenfassung – Ablauffehler

Zu optimistische Terminpläne	Fuzzy Front End	Überhastete Annäherung
Unzureichendes Risikomanagement	Streichung von Entwicklungsschritten	Vernachlässigung wichtiger Aufgaben
Lieferantenfehler	Unzureichendes Design	Planen zum späteren Auffangen
Unzureichende Planung	Streichung der Qualitätszusicherung	Code-like-hell Programmierung
Planungsabbruch unter Druck	Unzureichendes Management	

„Klassische“ Fehler
Zusammenfassung – Produkt und Technologie

Produktfehler		Technologiefehler	
Goldgepresste Bedürfnisse	Drück mich – Zieh mich Negierung	Silberkugel Syndrom	Mangel an automatischer SCC
Schleichendes Merkmal	Forschungsorientierte Entwicklung	Überschätzte Einsparung durch neue Praktiken	
Goldgepresste Entwickler		Toolwechsel während des Projektes	



Quelle: Steven McConnell „Rapid Development“ Seite 39

Gilligan‘s Insel

- Verhinderung der Fehler ist wie die Flucht von Gilligan‘s Insel
- Am Anfang scheint es zu funktionieren
- Am Ende ist man wieder am Anfang

Der Ausweg

- Aufschreibung der schlimmsten Fehler der letzten Projekte
- Diese Liste als Start nehmen
- Austausch mit Kollegen in anderen Firmen
- Liste öffentlich machen

Quellen

- Steve McConnell
  „Rapid Development“