Entwicklungsmethoden

Prof. Dr. Josef M. Joller
jjoller@hsr.ch

DER SOFTWARE PROZESS

Übersicht

Kunden, Entwickler und Benutzer

Anforderungsphase

Spezifikationsphase

Designphase

Implementationsphase

Integrationsphase

Wartungsphase

Ablösung

Inhärente Probleme der Softwareproduktion

Der Software Prozess

Life-Cycle Modell : Technik (Methode)

CASE Tool : Tools

Individuun : Teams

Auf diesen drei Fundamenten ruhen Software Projekte

Die drei Säulen müssen ausbalanciert vorhanden sein

- Werkzeuge in den ungeübten Händen sind nutzlos
- Methoden, die man nicht versteht, funktionieren nicht

Gibt es eine Testphase?

Es gibt keine Testphase!

Tests müssen laufend durchgeführt werden (TQM)

Verifikation

- wird nach Abschluss des Projektes drchgeführt

Validation

- entspricht das Produkt den Anforderungen des Kunden

Dokumentationsphase

Es gibt keine Dokumentationsphase

In jeder Phase müssen die erzielten Ergebnisse dokumentiert werden

- die Nachdokumentation funktioniert nicht
- das Produkt verändert sich laufend - die Änderungen müssen dokumentiert werden

Anforderungsphase

Der Entwicklungsprozess beginnt, wenn der Kunde die Entwicklung kontaktiert, um Software zu installieren oder erstellen, mit deren Hilfe der Kunde einen wirtschaftlichen Nutzen erzielen kann.

Annahme

- es ist möglich die neue Software kostenmässig zu rechtfertigen

Erarbeiten eines Verständnisses der Kundenanforderungen

- was benötigt der Kunde, nicht was möchte er!

Moving Target Problem

- die Firma ändert sich im Verlaufe des Projekts

Testen während der Anforderungsphase

Rapid Prototyping

- überprüfen Sie den Prototypen sehr genau!
- versichern Sie sich, dass das Produkt dem entspricht, was der Kunde benötigt
- bemühen Sie sich darum, das Produkt korrekt abzuliefern.

Dokumentation / Resultat

- Prototype zeigt, wie das Produkt aussehen könnte
- Das Anforderungsdokument hält die Anforderungen fest und basiert teils auf dem Prototypen.

Spezifikations Phase

Spezifikations Dokument ("Spezifikationen")

- Das Spezifikations-Dokument wird vom Spezifikationsteam erstellt.
- Es handelt sich um ein "legales" Dokument
- Das Dokument sollte keine Redewendungen wie "optimal", "98% vollständig" ... enthalten
- beschreibt, WAS geliefert werden muss
- Das Dokument muss von den Schlüsselpersonen unterschrieben werden.
- Das Dokument bildet die Basis für den Software Entwicklungsplan

Darf nicht

- mehrdeutig
- unvollständig
- widersprüchlich sein

Testen in der Spezifikationsphase

Verfolgbarkeit

- Aussagen in der Spezifikation sollten den Aussagen und Anforderungen des Kunden zugeordnet werden können.

Review

- das Spezifikationsteam und der Kunde müssen zusammen ein Review durchführen.
- Ziel des Reviews ist es, zu prüfen, ob die Spezifikation korrekt ist

SPMP

- ist es möglich, mehrere unabhängige Aufwandschätzungen zu erhalten?

Dokumentation der Spezifikationsphase

Spezifikation (das Dokument)

- beschreibt im Detail, was das Produkt kann
- enthält eine Beschreibung allfälliger Beschränkungen des Produkts
- beschreibt Eingaben und Ausgaben des Dokuments

SPMP

- der Softwareplan kann aufbauend auf einem Standard definiert werden
  - MIL Standard
  - IEEE Standard
  - Formenstandard

Design Phase

Spezifikation—was

Design—wie

Festhalten von Entscheiden in der Designphase

- Dokumentation allfälliger Sackgassen
- Hinweise für das Wartungsteam
- Idealerweise ist das Design “offen”

Architektur Design

- zerlegen des Systems in Module

Detail Design

- Design der einzelnen Module:
  - Algorithmen und Datenstrukturen

Testen in der Design Phase

Traceability

- auch hier: kann man Designgrössen den Anforderungen des Kunden zuordnen?

Review

- wird der Kunde das erhalten, was er benötigt?
- ABER:
  - da die Design-Dokumente in der Regel sehr technisch sind, wird der Kunde kaum anwesend sein.
- Mögliche Fehler
  - logische Fehler, Interface Fehler, Abweischungen von den Anforderungen

Implementations Phase

Implementieren des Designs

- programmieren der Module aus dem Design

Testen

- Review
  - auch hier!
- Testfälle
  - von Hand durchspielen (am Schreibtisch)
  - formales Testen durch die Software Qualitätskontrolle (SQA)

Dokumentation

- Quellcode
  - Testfälle, mit erwarteten Ausgaben

Integrations Phase

Kombination der Module zu einem Produkt

- Die Integrationsreihenfolge kann sein:
  - Top down
    - dabei sind die Designfehler sofort zu erkennen
  - Bottom up
    - Designfehler werden erst sehr spät erkennbar

Testen

- Produkttests
  - Integrationstests
  - prüfen, ob das Zusammenspiel der Module funktioniert
  - prüfen der Modulinterfaces
- Akzeptanztests
  - der Kunde testet das Produkt mit seinen Daten und prüft, ob seine Anforderungen erfüllt werden.

Dokumentation

- kommentierter Quellcode
- Testfälle für das System als Ganzes.

Wartungs Phase

Wartung

- umfasst alle Änderungen an der Software nach Ablieferung an den Kunden
- die Phase ist die teuerste aller Phasen
- Dokumentationen fehlen oft

Testen

- Regression Tests
  - Tests müssen bei Änderungen wiederholt werden.

Dokumentation

- alle Änderungen müssen dokumentiert werden.
- bereits durchgeführte Tests müssen wiederholt werden.

Ablösung

Gure Software wird gewartet, nicht abgelöst!

Software kann auch völlig neu geschrieben werden

- Software kann unwartbar sein
  - weil sich das Design wesentlich verändert hat
  - das System wird völlig neu, auf unterschiedlicher HW/SW implementiert
  - Dokumentationen liegen eventuell nicht mehr vor!
  - HW / Systemsoftware Änderungen könnten zu Anpassungen führen, welche eine Neuentwicklung nötig machen

Das Ablösen von Software ist selten.

Prozess-Spezifische Schwierigkeiten

Entspricht das Produkt den Anforderungen des Kunden?

Ist die Spezifikation wiederspruchtsfrei oder fehlen wichtige Teile

Inhärente Probleme der Software Produktion

Hardware besitzt inhärente Grenzen

Software ebenfalls!

Es gibt keine Silver Bullet (Kugel, mit der man den Werwolf erledigen kann) wegen:

- Komplexität der Software
- Konformität
- Änderbarkeit der Software
- Unsichtbarkeit der Software

Gemäss Boehm gibt es

- Probleme, die SW inhärent sind und nicht geändert werden können
- Probleme, die zufällig sind und mit neuer Technik reduziert werden können.

Komplexität

Software ist viel komplexer als Hardware

- bei der Erstellung der Software wird ein hohes Abstraktionsvermögen benötigt
- man kann die Software nicht verstehen, wenn man sie als Ganzes nicht versteht (und das Ganze SW Paket kann sehr umfangreich sein. Beispiel: ein SAP Parameter kann sich global auf die Funktionsweise von SAP auswirken).
- Das Management von SW Projekten ist äusserst schwierig
- Die Wartung der SW ist lomplex
  die Dokumentation auch (sofern sie aktuell gehalten werden soll)

Konformität ist nicht eindeutig

Type 1 Konformität: konform zu bestehenden Abläufen

- Automatisierung der Abläufe einer bestens funktionierenden Firma
  - in diesem Fall muss die Software den bestehenden Process abbilden
  - die Software muss nicht neue Konzepte beinhalten, sie muss einfach den bestehenden Prozess korrekt implementieren (Konformität : so wie heute)

Type 2 Konformität: konform geplanten Abläufen

- Steuerung der Prozesse für eine geplante Firma
  - die Software kann innovative Ideen implementieren
  - Konformität heisst: so wie geplant

Änderbarkeit

Software ist leichter zu ändern als Hardware

Warum wird SW verändert?

- Realität erfordert Änderungen
- Die Software soll neuen Gegebenheiten angepasst werden
- SW lässt sich leichter ändern als beispielsweise die HW oder die Leute (eines Prozesses)
- Software lebt lange (~15 Jahre) verglichen mit der Hardware (~4 Jahre) [Ausnahme: MS basierte SW bedingt HW Änderungen bei der Installation neuer SW]

Unsichtbarkeit

Software ist unsichtbar und kann auch nicht einfach visualisiert werden.

Es ist sehr schwierig die gesamte Komplexität der SW zu erfassen

Eine Teilsicht der SW kann zu Fehlschlüssen führen: Sie müssen die SW als Ganzes zumindest grob kennen.

Gibt es eine Silver Bullet?

Mögliche Ansätze

- High-level Sprachen (Skriptsprachen, 4GL, ..)
- moderne Betriebsysteme gestatten leichtere Migration und Portabilität und damit eine längere und umfassendere Nutzung
- CASE Tools

Alle diese Ansätze reichen nicht aus! Es gibt intrinsische Probleme

In den letzten Jahren haben Fortschritte erziehlt:

- 6% jährliche Zunahme der Produktivität

Aber keine massive Steigerung (keine “Silver Bullet”)

Verbesserungen des SW Prozesses

U.S. Department of Defense startete mehrere Initiativen

Beispiel: das Software Engineering Institute (SEI) an der Carnegie Mellon University in Pttsburgh

Ein grundlegendes Problem in der SW Entwicklung

- der Software Prozess ist immer noch schlecht bekannt (inhärente Probleme)

SEI und andere SW Prozess-Initiativen

- Capability Maturity Model (CMM)
  SEI (Software Engineering Institute @ CMU)
- ISO 9000-Serie
- ISO/IEC 15504

Capability Maturity Model

CMM ist kein Lebenszyklus Modell

CMM besteht aus Strategien für die Verbesserung des SW Prozesses

- SW–CMM für Software
- P–CMM für Human Resources (“Personen” : Teams)
- SE–CMM für Systems Engineering (Technik, Methoden)
- IPD–CMM für Integrierte Produkt Development
- SA–für Software Akquisition

Vereinheitlicht werden diese Strategien in CMMI (Capability Maturity Model Integration)

http://www.sei.cmu.edu/cmm/

SW–CMM

Eine Strategie für die Verbesserung des SW Prozesses

- 1986 vom SEI vorgeschlagen
- Grundlegende Idee:
  - ein verbesserter SW Prozess führt zu einer erhöhten SW Qualität und hoffentlich zu korrekter Planung (Budget und Zeit)
  - ein verbessertes SW Prozessmanagement führt zu verbesserten SW Entwicklungstechniken, verbesserten Ansätzen

CMM definiert fünf “Maturity” Levels

- jede Entwicklungsorganisation muss sich hoch arbeiten, von einem Level zum nächsten

Levels

Level 1 = Ad hoc Approach

- der gesamte Prozess ist nicht planbar
- Management befasst sich primär mit Feuerwehrübungen

Die meisten SW Entwicklungsfirmen sind auf Level 1

Beispiele (http://www.sei.cmu.edu/sema/pub_ml.html)

- Level 2
  - Bosch Japan
  - Hewlett Packard
  - Oerlikon Aerospace
- Level 3
  - Boeing
  - General Dynamics
- Level 4
  - Lockheed Martin Air Traffic Management
- Level 5
  - Boeing Defense & Space Group

Level 2. Repeatable Level

Grundlegende SW Management Prozesse sind klar definiert

- Management Entscheide werden auf Grund der Erfahrung aus früheren Projekten gemacht
- Es werden Messungen (“Metriken”) eingeführt
- Mit Hilfe der Messergebnisse können Kosten- und Aufwandschätzungen für die nächsten Projekte gemacht werden.
- Probleme werden identifiziert und korrekte Aktionen eingeleitet.

Level 3. Defined Level

Der Software Prozess ist vollständig dokumentiert

- Management und technische Aspekte sind klar definiert.
- Es wird kontinuierlich versucht Qualität, Produktivität ... zu verbessern.
- Mit Hilfe von Reviews wird versucht die SW Qualität laufend zu verbessern
- CASE Tools machen Sinn, da der Prozess klar definiert ist und die Reproduzierbarkeit wichtig ist.

Level 4. Managed Level / Level 5. Optimizing Level

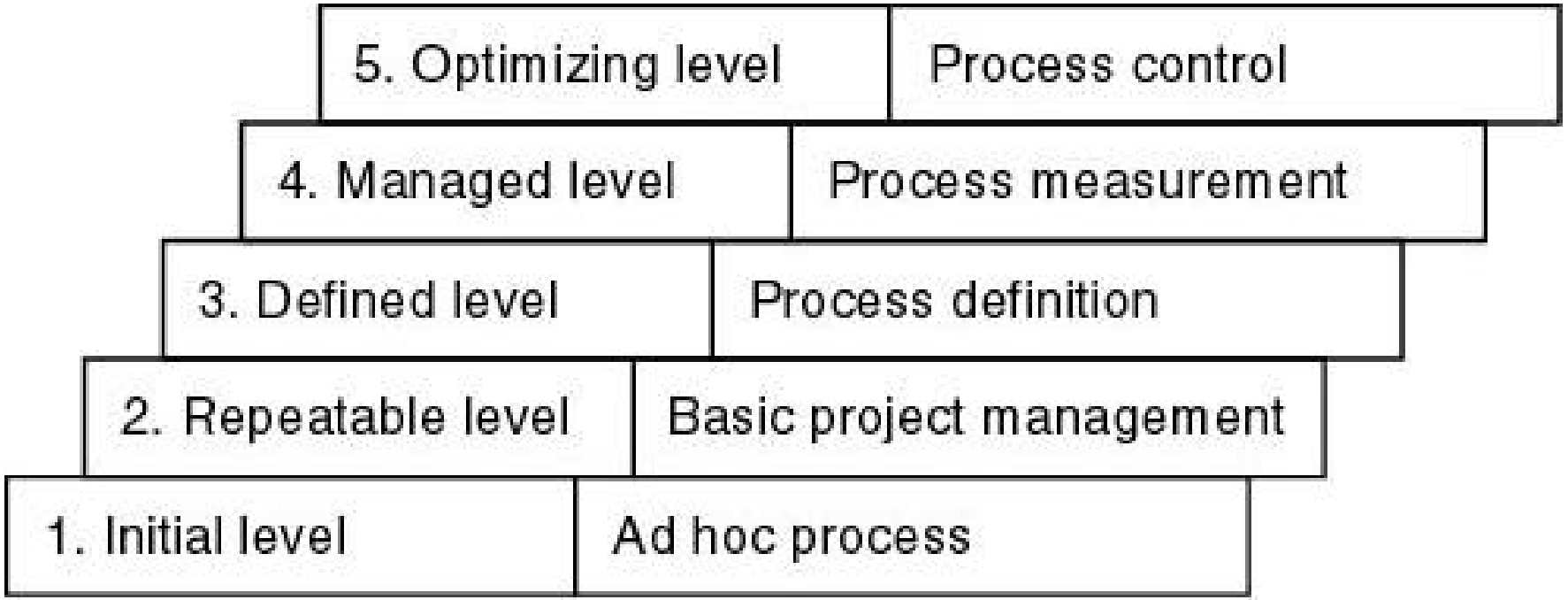
Level 4

- Für jedes Projekt werden Qualitäts- und Produktivitäts- Ziele definiert
- Qualität und Produktivität werden laufend überprüft
- statistische Qualitätskontrollen werden durchgeführt

Level 5

- jetzt versucht man den SW Prozess laufend zu verbessern
- statistische Qualitäts- und Prozess-Kontrollen sind eingeführt
- aus jedem Projekt werden Lehren für die nächsten Projekte gesammelt und dokumentiert

Zusammenfassung



Key Process Areas

Für jeden CMM Level wurden “key process areas” (KPAs) definiert

Level 2 KPAs :

- Requirements Management
- Projektplanung
- Projekttracking
- Konfigurationsmanagement
- Qualitätskontrolle

Unterschiede der Levels

- Level 2: Feststellen und korrigieren von Fehlern
- Level 5: Vermeiden von Fehlern

Erfahrungswerte

Eine Firma benötigt:

- 3 bis 5 Jahre, um von Level 1 auf Level 2 zu gelangen
- 1.5 bis 3 Jahre, um von Level 2 auf Level 3 zu klettern
- SEI Fragebogen zeigen Schwächen auf und schlagen mögliche Verbesserungen des Prozesses vor

Ursprünglliche Idee:

Departement of Defense bevorzugt Firmen mit höherem Level

Profitabilität

- Hughes Aircraft (Fullerton, CA)
  - Aufwand $500K (1987–90)
  - Savings: $2M /Jahr (Level 2 auf Level 3)
- Raytheon : Level 1 1988, Level 3 1993
  - Productivität verdoppelt
  - Return von $7.70 pro investiertem Dollar

ISO 9000

Besteht aus mehreren Standards für industrielle Aktivitäten

- ISO 9001 für Qualitätssysteme
- ISO 9000-3, Guidelines für die Anwendung von ISO 9001 auf Software
- CMM hat andere Ziele als ISO
  - ISO verlangt keine Prozessverbesserung
  - ISO legt viel Wert auf genaue und reproduzierbare Prozesse

- ISO 9000 wird in Geschäften mit der EU verlangt
- In den USA hat GE von Zulieferern ISO Zertifizierung verlangt
- Viele US Firmen sind ISO 9000 zertifiziert

ISO/IEC 15504

Software Process Improvement Capability dEtermination (SPICE)

- Internationale Prozessverbesserungsinitiative
- vom British Ministry of Defence (MOD) gestartet
- umfasst: Process Improvement, Software Procurement
- erweitert und verbessert CMM, ISO 9000
- SPICE ist ein Framework, nicht eine Methode
  - CMM, ISO 9000 sind SPICE konform
- Neue Bezeichnung ISO/IEC 15504 oder einfach 15504

Process Improvement Daten

Kategorie	Bereich	Mittelwert	Anzahl Messpunkte
Anzahl Jahre Software Prozess Improvement	1-9	3.5	24
jährliche Kosten pro SW Engineer	490-2004 USD	1375 USD	5
Produktivitätsgewinn pro Jahr	9-67%	35%	4
Früherkennungen von Fehlern (Verbesserung pro Jahr)	6-25%	22%	3
Verbesserung der Zeitplanung (pro Jahr)	15-23%	19%	2
Reduktion der Fehler nach SW Freigabe	10-94%	39%	5
Business Impact : Savings / Kosten der Verbesserungen	4 : 1 bis 8.8 : 1	5 : 1	5

SEI berichtete über 13 Organisationen in der ursprünglichen Studie

Process Improvement Daten

CMM Level	Anzahl Projekte	Durchlaufzeit Verbesserung	Anzahl Fehler (Entwicklungs-phase)	relative Produktivität
Level 1	3	1.0	-	-
Level 2	9	3.2	890	1
Level 3	5	2.7	411	0.8
Level 4	8	5	205	2.3
Level 5	9	7.8	126	2.8

Resultate von 34 Motorola Projekten