Todo list

1 Einführung

Das Hochhalten von Softwarequalität ist eine sehr anspruchsvolle Aufgabe, welche eine gewisse Planung vorraussetzt. Gerade bei Projekten mit hohen Budgets und materiellen Einsätzen werden hohe Anforderungen an das Qualitätsmanagement gesetzt.

Einige bekannte Fehlschläge in der Softwareentwicklung hätten wahrscheinlich mit besseren Qualitätssicherungsmaßnahmen verhindert werden können:

- Pioneer 4 verfehlte den Mond
- unnötige Mahnungen durch die französische Finanzverwaltung
- das Herausgeben von faulen Krediten, was schlussendlich zum Bankencrash geführt hat
- Verlust einer Segelyacht im Pazifik

1.1 Verifikation und Validierung

Die Begriffe sollten klar getrennt werden, da sie nicht synonym verwendet werden können, und bei Softwareprozessen und Softwarequalität eine gewichtige Rolle spielen.

Die Verifikation stellt fest ob die Software mit der vorhandenen Spezifikation übereinstimmt, wohingegen die Validierung bestimmt ob die Software für den Kunden auch wirklich nützlich ist.

1.2 Technical Debt

Technical Debts sind ein wichtiges Thema in der Qualitätssicherung. Der Begriff wurde aus dem Finanzwesen übernommen (Debt = Schulden). In der Softwareentwicklungen ist damit gemeint, dass ungeschickte Lösungen irgendwann gefixt werden müssen.

Diese technische Schulden kann man bewusst eingehen, um Termine zu halten. Allerdings muss man immer im Hinterkopf behalten, dass man diese Schulden zu einem späteren Zeitpunkt auch wieder bezahlen muss.

Einer der größten Unterschiede zwischen klassischen und agilen Projektmanagementmethoden ist der Umgang mit diesen technischen Schulden. Bei agilen Methoden werden diese Schulden bewusst eingegangen, um ein schnelleres iteratives Vorgehen zu gewährleisten. Bei diesen Methodiken ist allerdings auch die Gefahr einer Überschuldung ungleich höher als bei den klassichen.

Allerdings werden in der Regel auch bei klassischen Methoden technische Schulden verursacht, nämlich in Form von Änderungen in den Anforderungen, auf welche in agilen Methoden besser reagiert werden kann.

2 Hausaufgabe Besprechnung

Es sollte klar definiert werden was in den Sonderfällen passiert bzw. was in diesen Fällen zurückgegeben wird (z.B. komplexe Zahlen da in Wurzel negative Zahl). Gleitkommzahlen sind "Teufelswerk"! Wenn man zwei gleiche Zahlen (die berechnet wurden aber unterschiedliche Rundungsfehler haben) voneinander voneinander subtrahiert bleibt nicht 0 übrig sondern der Rundungsfehler. Durch Rundungsfehler können somit 2 verschiedene Ergebnisse entstehen die eigentlich eines wären - darum könnte man anstatt der Mitternachtsformel eine andere Formel / einen anderen Ansatz verwenden (z.B. Newton-Näherungsverfahren).

> Äls Softwareentwickler ist es durchaus erlaubt sein Hirn einzuschalten! - Gleitkommzahlen sind Teufelswerk!"

3 Wichtige Begriffe

- Softwareprozesse (Abfolge von Tätigkeiten, durch die ein Software-Produkt entsteht)
- Vorgehensmodell (Vereinfachte Beschreibung eines Softwareprozesses)
- Methode (Strukturierter Ansatz für die Software-Entwicklung)

3.1 Software-Engineering

Software-Engineering ist eine technische Disziplin welche eine Lösung für den Anwender bieten will unter Einsatz von Theorien, Methoden und Werkzeugen und Berücksichtigung von Organisation, Management und Entwicklung.

3.2 Informatik

Grundlage der Wissenschaft für Software-Ingeneure (Theorie, Konzepte, ...).

3.3 ESSENCE Kernel Overview

In diesem Model gibt es verschiedene Zustände für die Anforderungen und jeder dieser Zustände hat selbst wieder Kriterien welche einem dabei helfen in welchem Zustein ein Projekt ist.

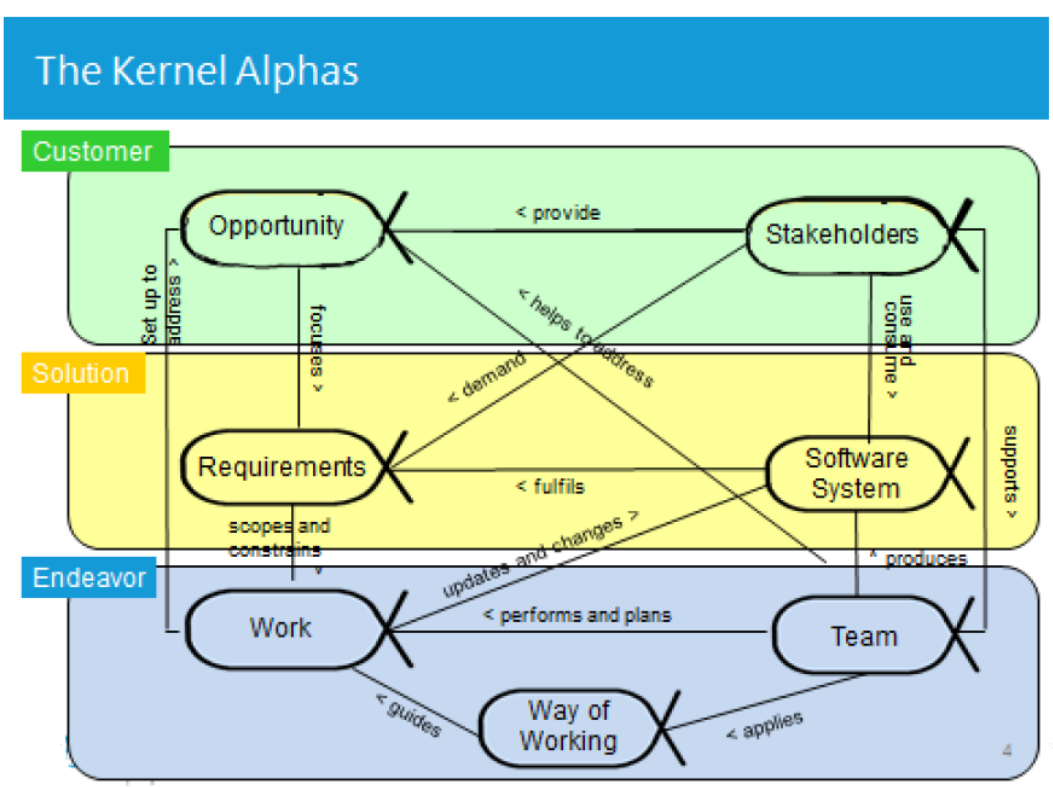

Abbildung 3.1: ESSENCE Kernel Overview

Zustände:

- Conceived
- Bounded
- Coherent
- Accepted
- Adressed
- Fulfilled

3.4 Qualität

Grad in dem die inhärenten Eigenschaften des Produkts Anforderungen erfüllen.

- **Explizite Anforderungen**
- **Implizite Anforderungen** - Anforderungen welche existieren aber den Stakeholdern nicht bewusst sind.
- **Nicht explizite Anforderungen** - Stakeholer wissen, dass sie diese Anforderungen gibt aber sie teilen diese nicht mit (sind quasis eh klar).
- **Objektiv** - Vollkommen klar, dass man etwas braucht.
- **Subjektiv** - Vermeintliche Anforderungen, welche nicht wirklich wichtig sind.
- alle betroffenen / interessierten Personen

3.4.1 Begriffsabgrenzung

Technisches computer-basiertes System System welches ausschließlich aus Soft- und Hardware-Komponenten besteht.

Soziotechnisches System System bestehend aus einem oder mehreren technischen Systemen, den Menschen die es bedienen, den notwendingen Arbeitsprozessen, organisatorischen Richtlinien, usw.

Systeme:

- haben systemspezifische Eigenschaften die nur dem System als Ganzem zugeordnet werden können
- sind häufig nicht deterministisch
- hängen von organisatorischen Zielen ab

3.5 Kritische Systeme

Relevante Systemeigenschaften für kritische Systeme sind:

- Reparierfähigkeit
- Wartbarkeit
- Überlebensfähigkeit
- Fehlertoleranz

Kritisches System Systeme, bei dessen Ausfall oder Fehlfunktion großen Schaden anrichten kann (wirtschaftliche Verluste, physische Schäden, Gefahr für Gesundheit und Leben von Menschen).

Sicherheitskritisches System Schäden an der Umwelt und/oder Gefahr für Gesundheit und Leben von Menschen (Bohrinsel im Golf von Mexiko).

Aufgabenkritisches System Aufgaben die ein System erledigen solle werden nicht durchgeführt (z.B. Bank).

Geschäftskritisches System Extrem hohe Kosten bzw. signifikante Gewinnausfälle können die Folge eines Systemausfalls sein.

t a stuck

3.6 Sammlung von Anforderungen

3.6.1 Interviews

Technik

- Interviews mit Fragebogen
- Geschlossene Interviews vorgegebener Fragebogen abarbeiten
- Offene Interviews mit einer Diskussion zwischen Analyseteam und Beteiligten

Durchführungsempfehlung

- Mischung aller Methoden
- Vorbereitung aller Diskussionen als geschlossenes Interview
- gut und interessiert zuhören
- gezielt Fragen

Probleme

- Jargon des Anwendungsgebiets
- Implizites Wissen

Eignung

- Verständnis der Benutzeranforderungen
- Ergänzung zu anderen Informationsquellen mit zum Beispiel Dokumentationen oder Beobachtungen

Szenarien

- Grundbestandteile
  - Ausgangssituation
  - Ereignisablauf
  - Ausnahmen und ihre Behandlung
- Varianten
  - Ad-hoc
  - Formell

3.6.2 Ethnografische Methode (Völkerkunde)

Grundidee: Beobachten ohne einzugreifen

- Beobachten der alltäglichen Arbeit im normalen Umfeld
- Notieren von Auffälligen
- Diskussion mit Experten
- Ableitung der Anforderungen

Stärken

- Erkennen von impliziten Anforderungen
- Erkennen von nicht explizierten Anforderungen
- Erkennen von Abweichungen

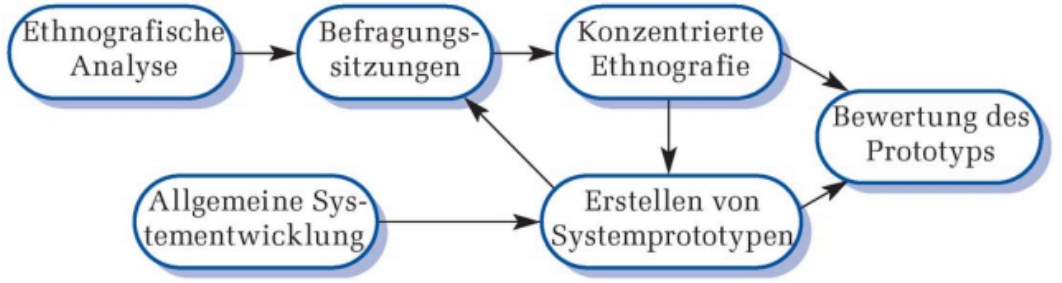

Abbildung 3.2: Verknüpfung von Ethografie und Prototypen nach Sommerville (Copyright Pearson Studium 2007)

3.7 Klassifizierung von Anforderungen

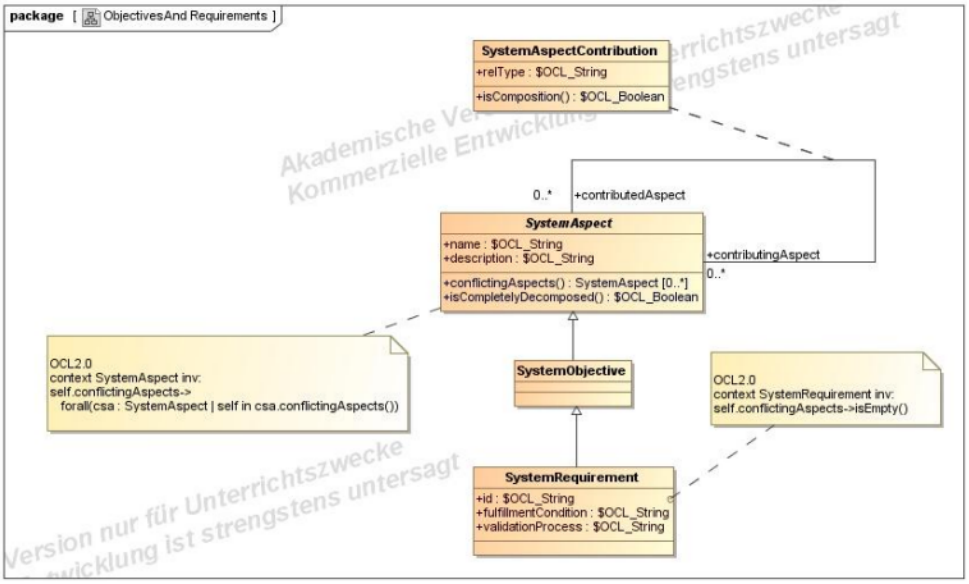

Abbildung 3.3: Metamodell

Aufgaben

- Erkennen von Duplikaten / Synonymen
- Beziehungen zwischen Anforderungen
- Gruppierung der Anforderungen

3.8 Validierung von Anforderungen

Wichtige Prüfungen sind:

- Gültigkeitsprüfung
- Konsistenzprüfung
- Vollständigkeitsprüfung (schwierig, aber mit etwas Bauchgefühl machbar)
- Realisierbarkeitsprüfung
- Verifizierbarkeitsprüfung

3.8.1 Techniken

Prototypen erstellen und Testfälle entwickeln. Falls das nicht möglich sind die Anforderugnen schlecht definiert.

3.8.2 Review

Teamzusammensetzung Es sollten Vertreter aller am Projekt beteiligten bzw. vom Projekt betroffenen Gruppen auf Anwenderseite, ausserdem Systemarchitekten und Vertreter der Softwareentwickler.

Durchführung Die Führung der Analyse liegt bei den jeweiligen Anwendervertretern. Diskutiert werden sollten alle Anforderungen. Dadurch können Fehler, Konflikte und Wiedersprüche aufgedekt werden. Das Ergebnis ist ein Review Bericht.

Prüfungen Die Konsistenz der Anforderungen sollten am Ende geprüft werden. Ausserdem sollte eine Vollständigkeitsprüfung durchgeführt werden.

3.8.3 Priorisierung von Anforderungen

Grundgedanke Phasenweise Implementierung der Software.

Prioritätsfestlegung

Teams festlegen, die nicht miteinander kommunizieren sollten. Ausserdem wird eine Bewertungsformel festgelegt.

Nutzwertanalyseteam Bewertet den Nutzen für die Anwender.

Kostenanalyseteam Schätzt die Kosten jeder Anforderung

Auswertung

Einflussgrößen

- Nutzwert jeder Anforderung
- Kosten jeder Anforderung
- Obergrenze des Aufwands für Stufe 1
- Abhängigkeiten

Ergebnis Liste der Anforderungen für nächste (bzw. erste) Ausbaustufe

3.9 Systemmodelle

Kontextmodelle Definiert die Systemgrenzen des Gesamtsystems und des technischen Systems. Das Gesamtsystem umfasst das Technische System und die Menschliche Komponente und definiert den Kontext. Das Technische System definiert den Scope.

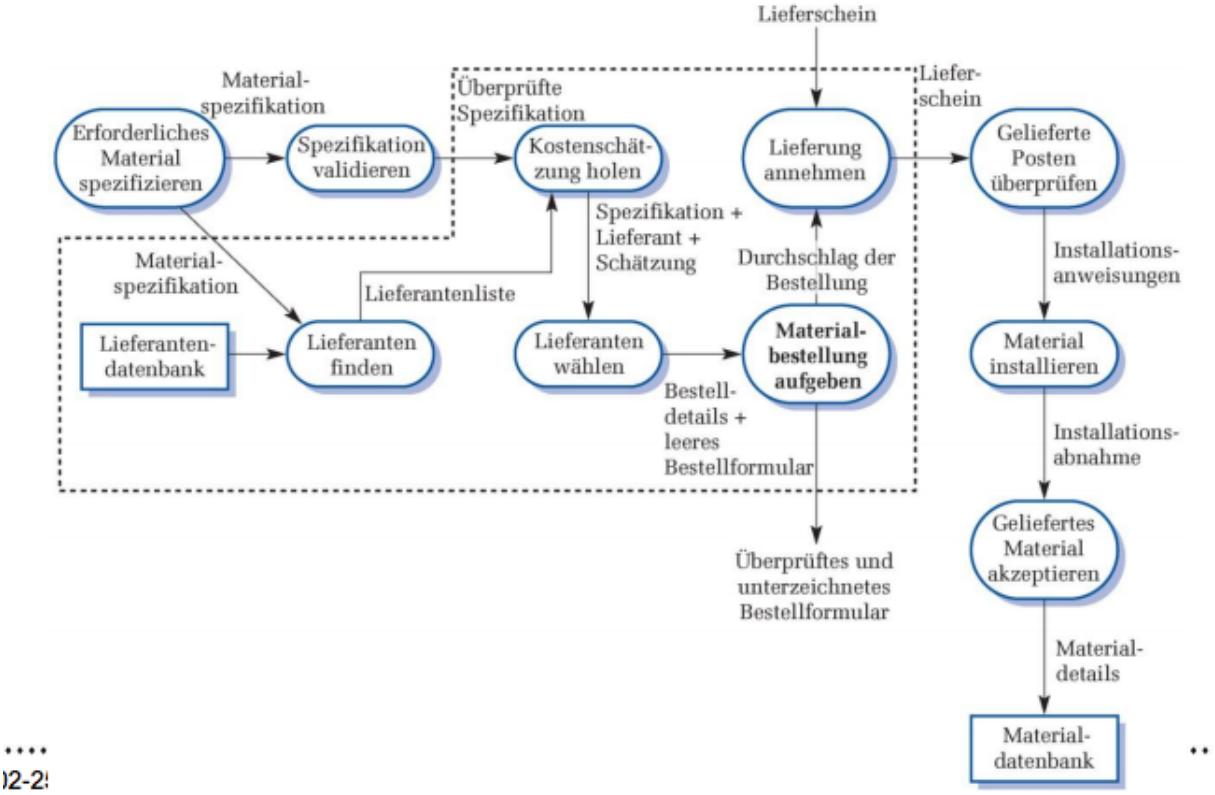

Abbildung 3.4: Kontextmodelle

Verhaltensmodelle Definiert die Abläufe in einem System. Sie können Datenfluss (Datenfocus) - oder Ereignis (Ereignisfocus) - Orientiert geschehen. Aussderm gibt es Mischformen davon. Dafür können Datenflussdiagramme (Abbildung 3.5), Zustandstabelld (Abbildung 3.6) und ähnliches verwendet werden.

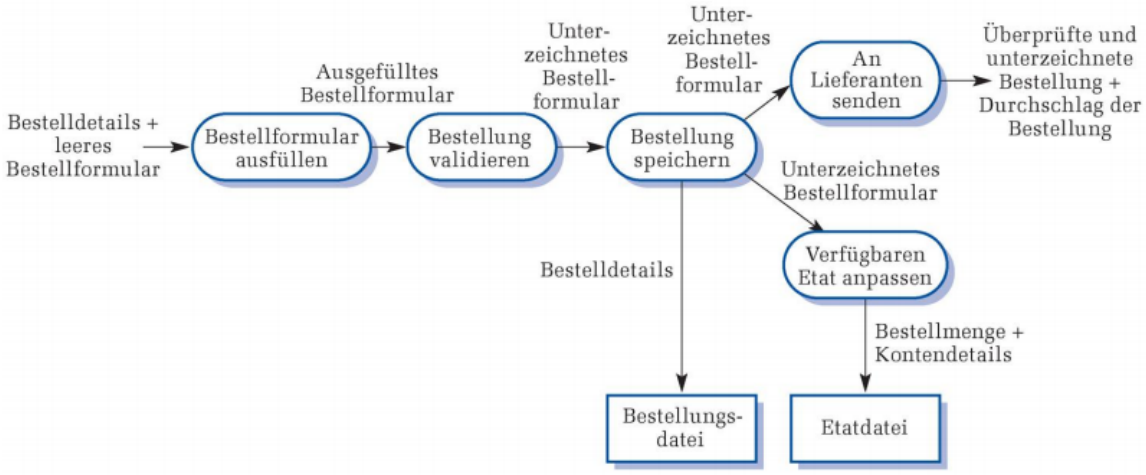

Abbildung 3.5: Datenflussdiagramm

Datenmodelle Definiert die logische und persistente (lokal, übergreifend mttels Datenbank und Datenaustausch) Datenstruktur. Dazu kann ER-, UML-Diagramme, OWL oder andere Ontologie-Darstellungen und andere.

Zustand	Beschreibung
Warten	Das Gerät wartet auf eine Eingabe. Die Anzeige zeigt die aktuelle Zeit an.
Halbe Leistung	Die Geräteleistung wird auf 300 Watt gesetzt. Die Anzeige zeigt „Halbe Leistung" an.
Volle Leistung	Die Geräteleistung wird auf 600 Watt gesetzt. Die Anzeige zeigt „Volle Leistung" an.
Zeiteinstellung	Die Garzeit wird auf die Eingabe des Benutzers gesetzt. Die Anzeige zeigt die gewählte Garzeit an und wird bei der Einstellung der Zeit aktualisiert.
Deaktiviert	Der Betrieb des Geräts ist aus Sicherheitsgründen deaktiviert. Das Licht im Gerät ist eingeschaltet. Die Anzeige zeigt „Nicht bereit" an.
Aktiviert	Der Betrieb des Geräts ist aktiviert. Das Licht im Gerät ist ausgeschaltet. Die Anzeige zeigt „Bereit" an.
Betrieb	Das Gerät ist in Betrieb. Das Licht im Gerät ist eingeschaltet. Die Anzeige zeigt den Countdown der Garzeit an. Nach dem Ende der Garzeit ertönt fünf Sekunden lang ein Signal. Das Licht im Gerät ist eingeschaltet. Die Anzeige zeigt „Kochvorgang beendet" an, während das Signal ertönt.
Stimulus	**Beschreibung**
Halbe Leistung	Der Benutzer hat die Taste „Halbe Leistung" betätigt.
Volle Leistung	Der Benutzer hat die Taste „Volle Leistung" betätigt.
Timer	Der Benutzer hat eine der Tasten für die Zeitschaltuhr betätigt.
Zahl	Der Benutzer hat eine Zifferntaste betätigt.
Tür offen	Die Gerätetür ist nicht geschlossen.
Tür geschlossen	Die Gerätetür ist geschlossen.
Start	Der Benutzer hat die Starttaste betätigt.
Abbruch	Der Benutzer hat die Abbruchtaste betätigt.

Abbildung 3.6: Zustandtabelle

Objektorientierte Modellierung Vereinigt die Funktionalität von Daten- und Verhaltensmodelle und können Daten, Datenflüsse, Datenstrukturen und Erreignise erfassen. Als Notation wird meist UML verwendet.

Sind die Klassen in dem Diagramm Abbildung 3.7 wirklich **Klassen oder Rollen**? Nein das sind eindeutig Rollen.

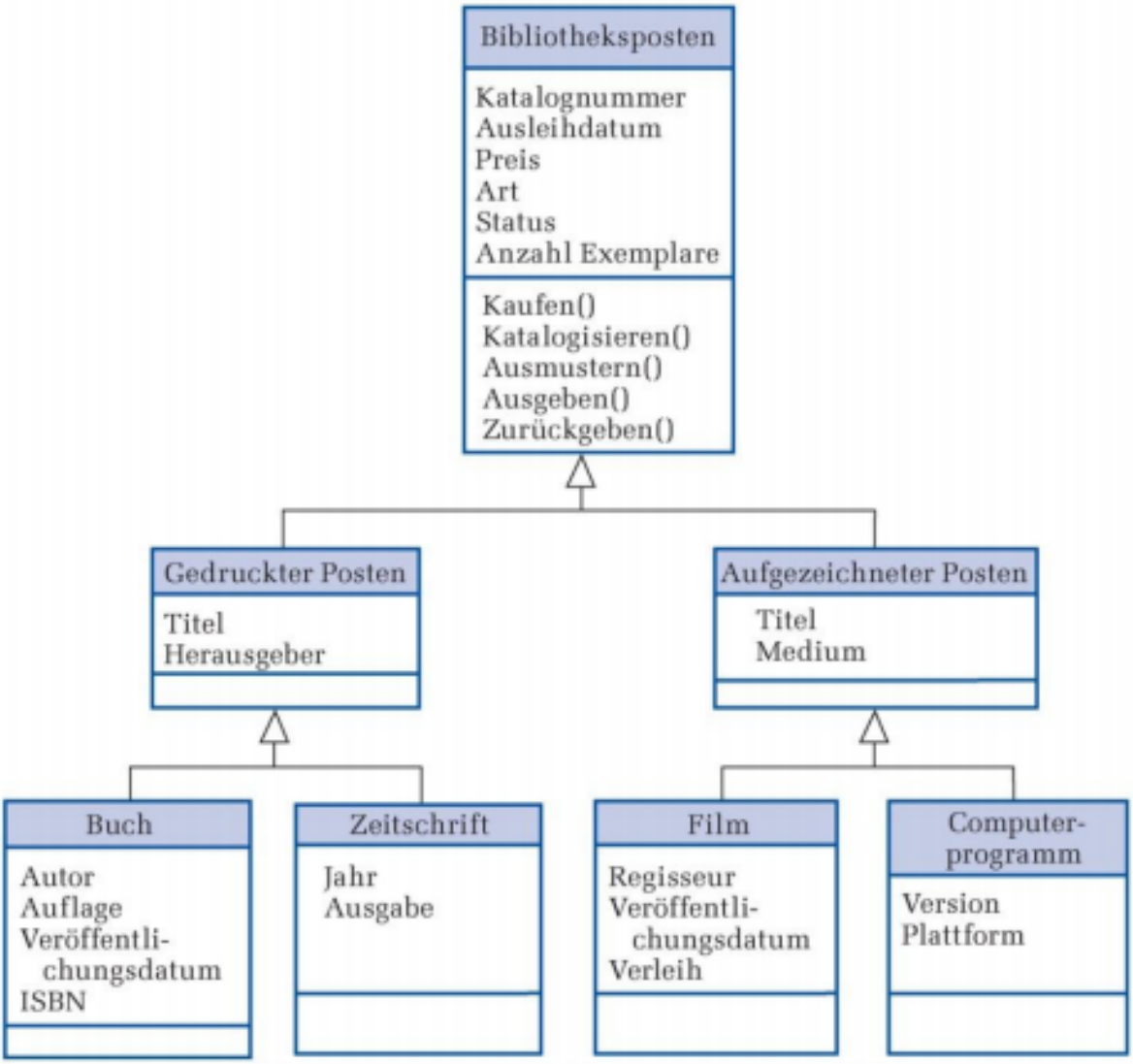

Abbildung 3.7: Objektorientierte Modellierung

Strukturierte Methoden Detailliert definierte Vorgehensweise bei der SW-Entwicklung. Normalerweise basierend auf einem Satz von Diagrammtypen. Definiert zusätzliche Regeln und Richtlinien. Beispiele dafür sind JSP, V-Modell oder RUP. Nachteile davon sind Mangelnde Unterstützung nicht-funktionaler Anforderungen und Anwendbarkeit für konkretes Problem oft schwer zu entscheiden.

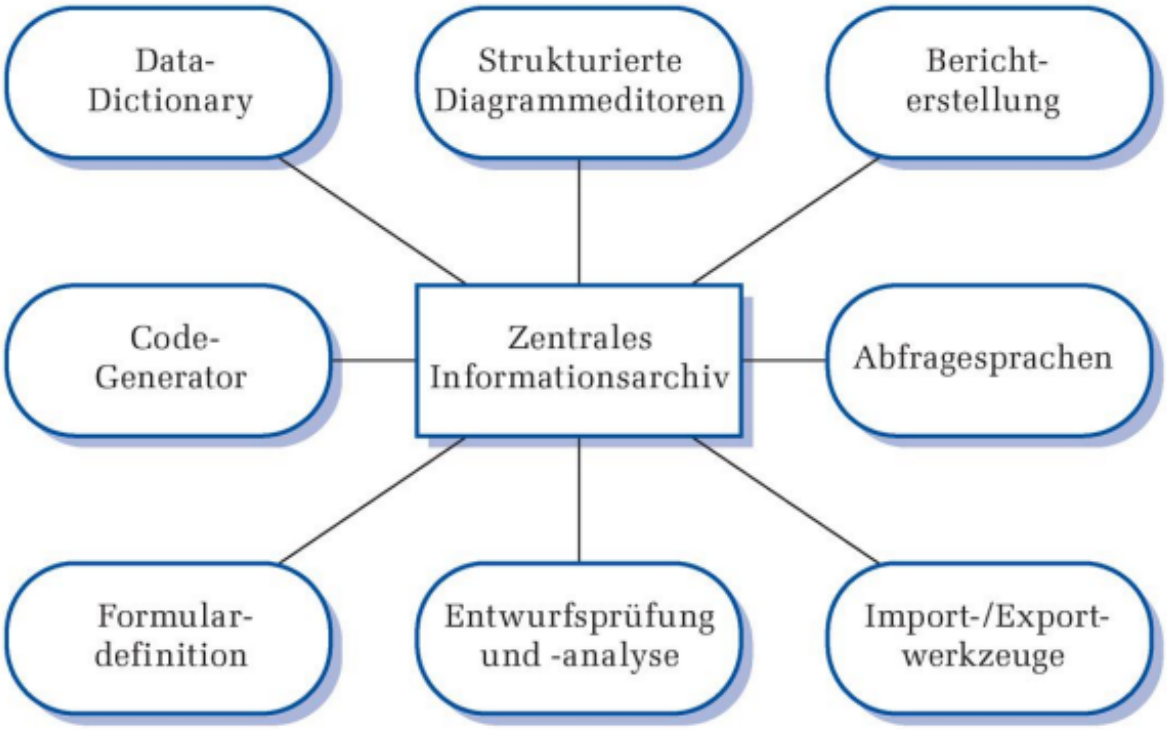

Abbildung 3.8: Strukturierte Methoden