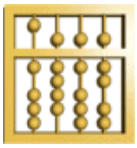
# Software Engineering:
# Software im Automobil

## 5./6. Vorlesung:
## Entwurfsprozess/Management und Organisation

Manfred Broy

Technische Universität München
Institut für Informatik
D-80290 München, Germany



## Vorgehensmodelle im Software und Systems Engineering

- Vorgehensmodelle bestimmen das Vorgehen (Planung und Durchführung) im Software und Systems Engineering
- Vorgehensmodelle sind in der Regel in Phasen organisiert
- Vorgehensmodelle zerfallen in
  - Prozessmodelle: Struktur der Aktivitäten
  - Produktmodelle: Struktur der Arbeitsprodukte
- Vorgehensmodelle helfen gleichermaßen den Entwicklern und den Projektmanagern
- Vorgehensmodelle standardisieren das Vorgehen und befördern ein einheitliches Verständnis
- Vorgehensmodelle bilden die Schnittstelle zwischen Projektorganisation und -management und den technischen Entwicklungstätigkeiten

- Wohldefinierte Vorgehensweise („Entwicklungsprozess") und Festlegung der zu erarbeitenden Ergebnisse („Entwicklungsprodukte")
- Klare Aufgaben und Rollenteilung
- Schrittweise Erarbeitung aller Anforderungen und Entwurfsfestlegungen („Lösungen") für das Zielsystem
- Darstellung wesentlicher Systemeigenschaften mit standardisierten Beschreibungsmitteln
- Integrierte konstruktive und analytische Qualitätssicherung
- Weitgehende, abgestimmte Werkzeugunterstützung



# Einordnung in Phasenmodell

- Das Phasenmodell (Vorgehensmodell) dient der Strukturierung des Vorgehens in Phasen unter Einbeziehung von Aufgaben der Projektorganisation und des Projektmanagements (Prozess- und Produktmodel: Entwickeln im Großen) .
- Produktmodell: Menge aller Entwicklungsprodukte und ihre Beziehungen
- Prozessmodell: Menge aller Entwicklungsaktivitäten
- Die Gesamtheit aller Entwicklungsschritte und -produkte ist in einem Vorgehensmodell zusammengefasst.
- Entwicklungsmethoden dienen als Verfahren oder Anleitungen zur Durchführung von einzelnen Entwicklungsschritten.
- Entwicklungsmethoden zielen auf einzelne Erarbeitungsschritte hin und ihre konkrete Durchführung.
- Bestimmte Entwicklungsmethoden können in unterschiedlichen Phasen anwendbar sein.

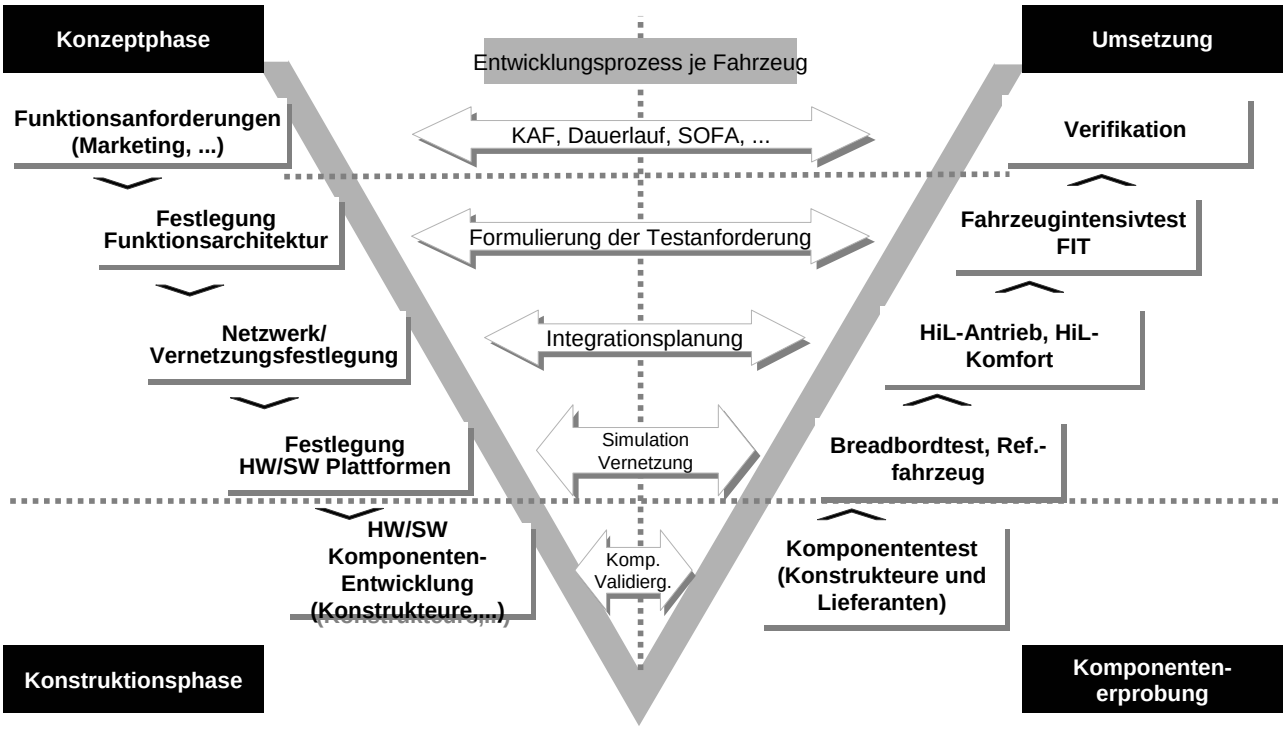
# Zur Beschreibung von Entwicklungsprodukten

- Entwicklungsprodukte sind in der Regel Protokolle, Ausarbeitungen, Berichte, Systembeschreibungen, Prototypen, Simulationsergebnisse, Erfahrungsdaten.
- In der Regel entsteht eine große Menge von Daten und Dokumenten im Rahmen einer System/Softwareentwicklung.

  Entwicklungsprodukte dokumentieren die Resultate der Entwicklungstätigkeit.
- Eine systematische Erzeugung, Qualitätssicherung und Verwaltung von Entwicklungsprodukten ist für umfangreichere Projekte unabdingbar.
- Der Fluss („Workflow") und die inhaltlichen Abhängigkeiten zwischen den Entwicklungsprodukten ist für das Verständnis des Vorgehens bestimmend.



# Aufgaben im Entwicklungsprozess



# Phasen Systementwicklung/Softwareentwicklung

- Konzeptfindung
- Anforderungen
  - ◇ Lastenheft
  - ◇ Pflichtenheft

Analyse

- Design
  - ◇ Systemarchitektur
  - ◇ Softwarearchitektur
    - Grobdesign
    - Feindesign

Entwurf

- Implementierung
- Test

Implementierung

- Integration
- Erprobung

Integration

- Einführung
- Produktion

Produktion

- Wartung

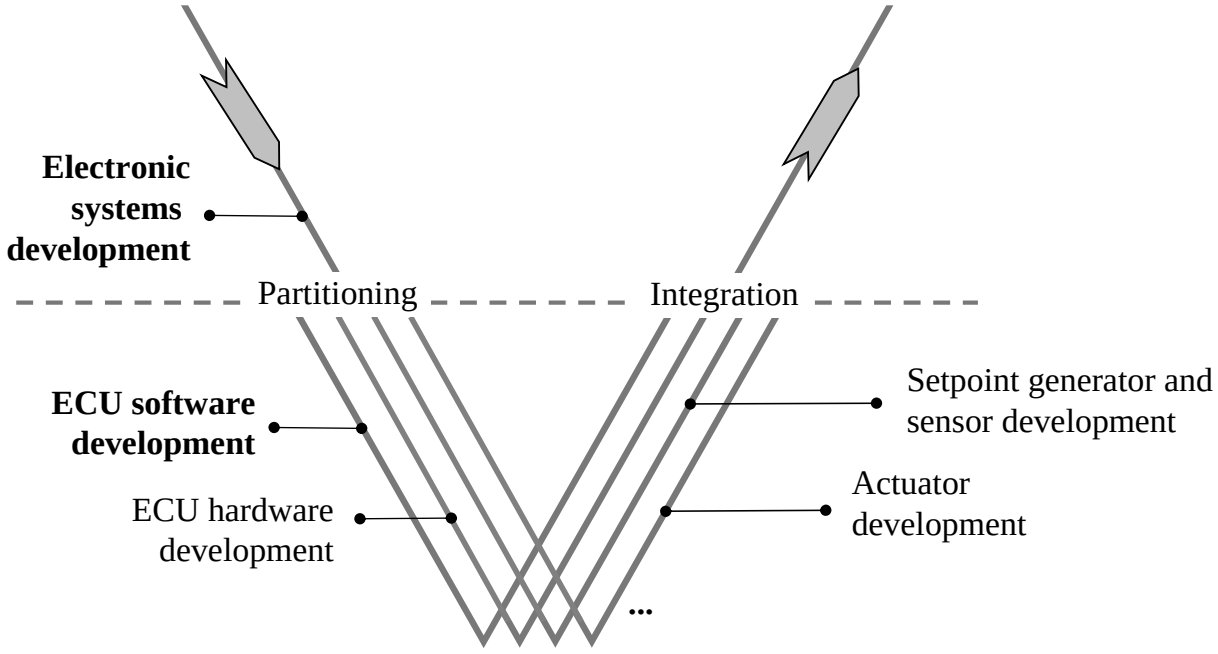
Betrieb



# Die Softwareentwicklung im PEP

- Software ist eingebettet ins Produkt
- Die Softwareentwicklung muss sich in den Produktentwicklungsprozess einfügen
- Somit gibt es vor- und nachgelagerte Prozesse in Hinblick auf die eigentliche Softwareentwicklung
- Die Entwicklungsprozesse für Produkt/System und Software weisen Ähnlichkeiten auf (Analyse/Architektur/Implementierung/Integration/Test), aber auch Besonderheiten

# Aufspaltung PEP in Teilentwicklungsprozesse






# Zeitverlauf PEP bis SOP (Start of Production)

Vorleistungsphase (SOP-60 Monate)

- Initialphase
- Konzeptphase (SOP-52 Monate)
- Vorbereitungsphase (SOP-36 Monate)

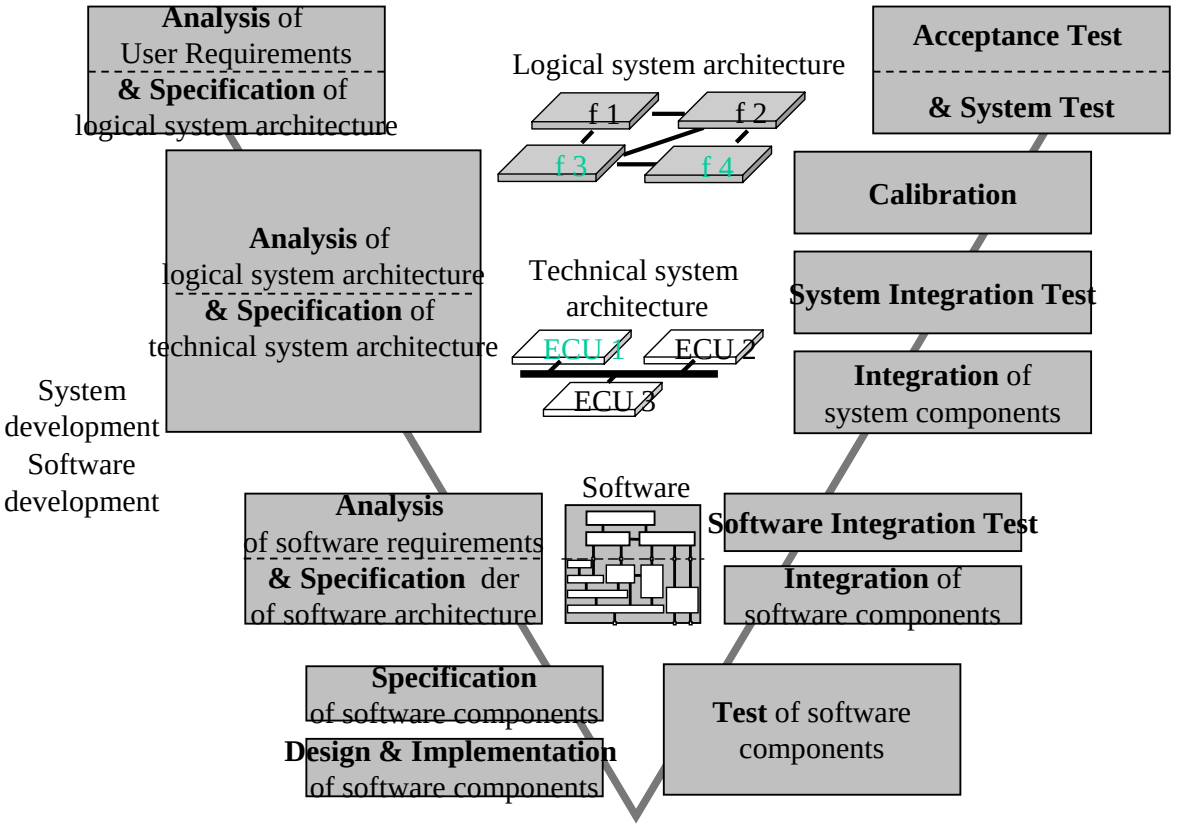
Serienentwicklung (SOP-30 Monate)

- Abstimmungsphase
- Bestätigungsphase (SOP-22 Monate)
- Reifephase (SOP-7 Monate)

Serie

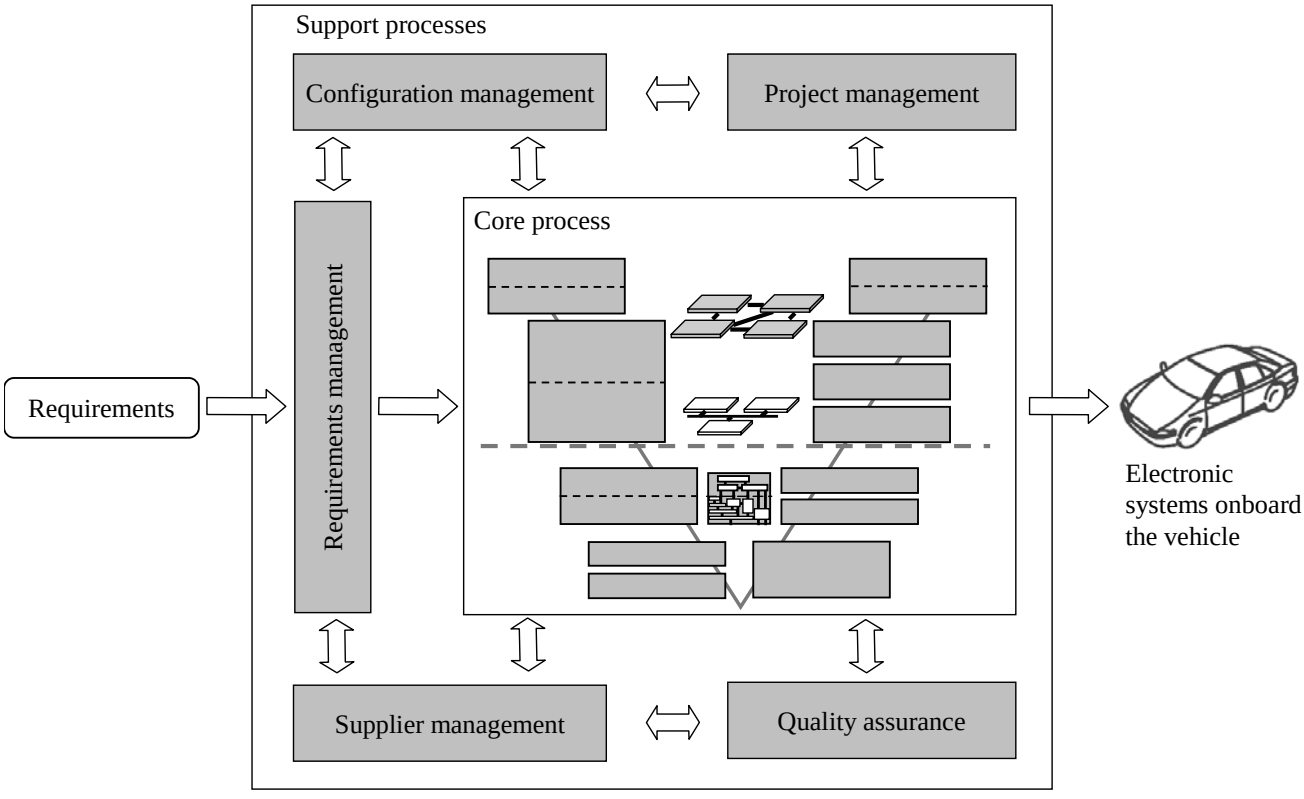
- Softwaremodifikation

# Aufspaltung PEP in Teilentwicklungsprozesse

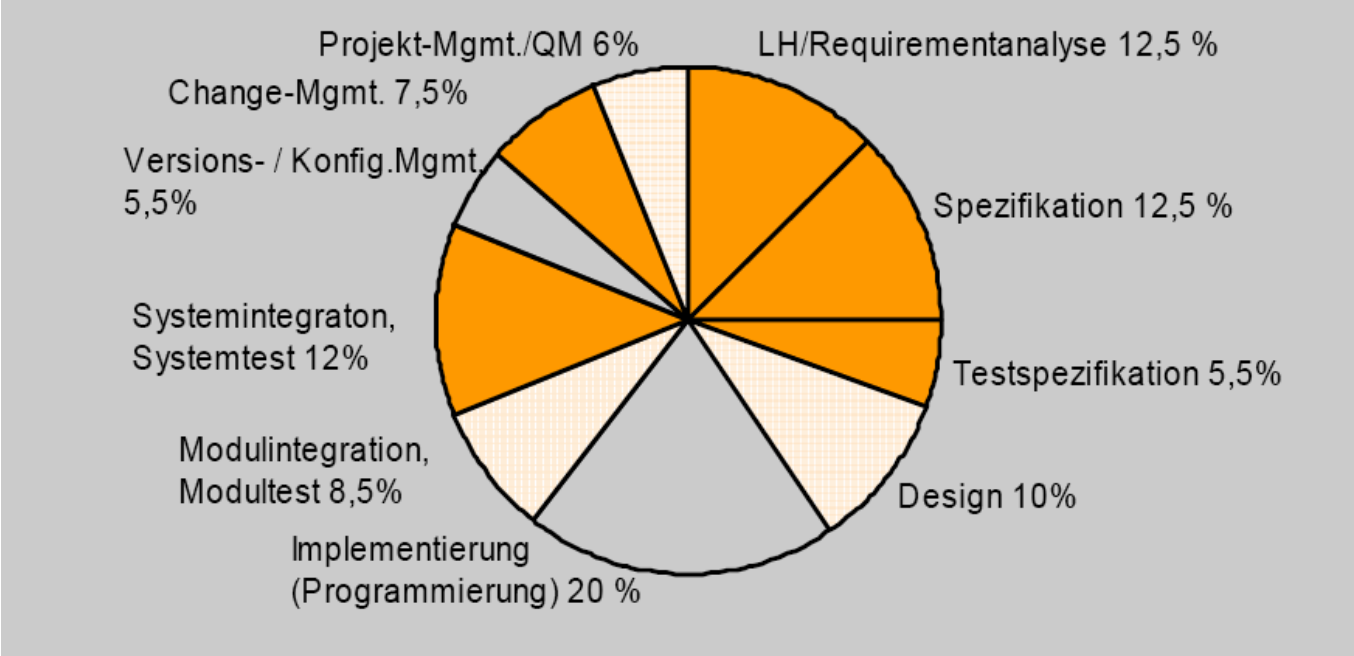





# Einbettung in Teilentwicklungsprozesse



# Aufwandsverteilung (Nach Audi/3soft)






# Sicherheitskritische Systeme

- Vorgaben IEC 61 508 beachten (Funktionale Sicherheit sicherheitsbezogener elektronischer/programmierbarer elektronischer Systeme, 2001)
- SIL Einstufung (Sicherheits-Integritätslevel)
- Gewährleistung der funktionalen Sicherheit
  - Projektmanagement
  - Qualitätssicherung

## Anforderungsanalyse/Requirements Engineering System

- In der Anforderungsanalyse werden die Anforderungen an ein IT-System festgelegt.
- Die Festlegung erfolgt zunächst in der Vorgabe entscheidender Ziele aus Sicht des Marketings oder der Gesamtsystemfunktionalität
- Diese allgemeinen Vorgaben und Ziele werden im Rahmen des RE in konkrete Vorgaben für den Aufbau und das Verhalten eines Systems umgesetzt, die als Vorgaben für den Entwurf und die Realisierung des Systems dienen.



## Bedeutung des Requirements Engineering

- Das Requirements Engineering wird oft unterschätzt, nicht sorgfältig genug durchgeführt oder ungenügend beherrscht
- Das Requirements Engineering bestimmt mit seinen Festlegungen entscheidend
  - ◇ die Funktionalität und damit
  - ◇ die Nützlichkeit und Komplexität

  eines Systems/Teilsystems
- Die Entwurfsräume sind im Automobil aufgrund der Flexibilität der eingebetteten Systeme enorm gewachsen. Deshalb herrscht hier mehr Entscheidungsfreiheit bei der Festlegung der Anforderungen.
- Fehlentscheidungen im Requirements Engineering belasten den ganzen folgenden Entwicklungsprozess und das Produkt im Feld (bis zu 50% von Kundenreklamationen sind auf ungenügendes Requirements Engineering zurückzuführen)

# Klassifizierung der Anforderungen

Es gibt eine Vielfalt unterschiedlichster Anforderungen:

- Funktionale Anforderungen:
  Funktionale Anforderungen betreffen das Verhalten des Systems.
- Nichtfunktionale Anforderungen:
  Nichtfunktionale Anforderungen betreffen Eigenschaften des Systems oder seiner Entwicklung, die sich nicht auf die Festlegung des detaillierten Verhalten ausrichten.

Im Requirements Engineering ist es nützlich, Anforderungen zu klassifizieren und so zu strukturieren, dass deutlich wird,

- Wie die Erfüllung einer Anforderungen überprüft wird
- In welchen Phasen des Entwicklungsprozess eine Anforderung berücksichtigt wird.

Stichworte: Requirements Verification, Requirements Tracing



# Prinzipien des Requirements Engineering

- Alle relevanten Informationsquellen und Interessengruppen („Stakeholder“) einbeziehen!
- Anforderungen sammeln, strukturieren, gruppieren, bewerten.
- Mögliche Alternativen sauber herausarbeiten, abwägen, Entscheidungskriterien festlegen und transparent entscheiden. Entscheidungsgründe dokumentieren!
- Anforderungen präzise dokumentieren.
- Details durch Modellbildung genau erfassen.
- Durch eine frühe Modellbildung werden Widersprüche, Mehrdeutigkeiten und Unvollständigkeiten in Anforderungen erkannt und können gezielt bereinigt werden.

# Funktionsnetze für Multifunktionssysteme

- In einem Funktionsnetz werden alle Funktionen eines Systems strukturiert zusammengefasst.
- Dies ergibt eine strukturierte Sicht auf die Systemfunktionen (Nutzungsfälle, „Use Cases“) und ihre Abhängigkeiten
- Typische Abhängigkeiten zwischen Systemfunktionen:
  - Eine Funktion stützt sich auf eine andere Funktion ab
  - Eine Funktion beeinflusst eine andere Funktion



# Nutzungsarchitektur: Funktionsnetz

Im Funktionsnetz wird die Gesamtfunktionalität der softwarebasierten Funktion beschrieben

- Welche Funktionen sollen vorhanden sein
- Wie ordnen sie sich unter (Funktionshierarchie)
- Welche Abhängigkeiten bestehen

# Funktionen

- Eine Systemfunktion (auch Dienst, Feature) ist eine Verhaltensweise des Systems oder eines Teilsystems zu einem bestimmten Zweck.
- Eine Funktion eines eingebetteten System kann als eine Menge von Interaktionsmustern (Signalfluss) verstanden werden. Dadurch werden Reaktionsweisen eines Systems auf gewisse Eingaben (Ereignisse) in der Systemumgebung beschrieben.



# Nutzungsfälle

- Ein Nutzungsfall beschreibt eine bestimmte Funktion, in der Regel exemplarisch, aus Sicht eines Nutzers.
- In einem Nutzungsfall werden alle an dem Fall beteiligte der Systemumgebung aufgelistet

  Beispiel: Fensterheber

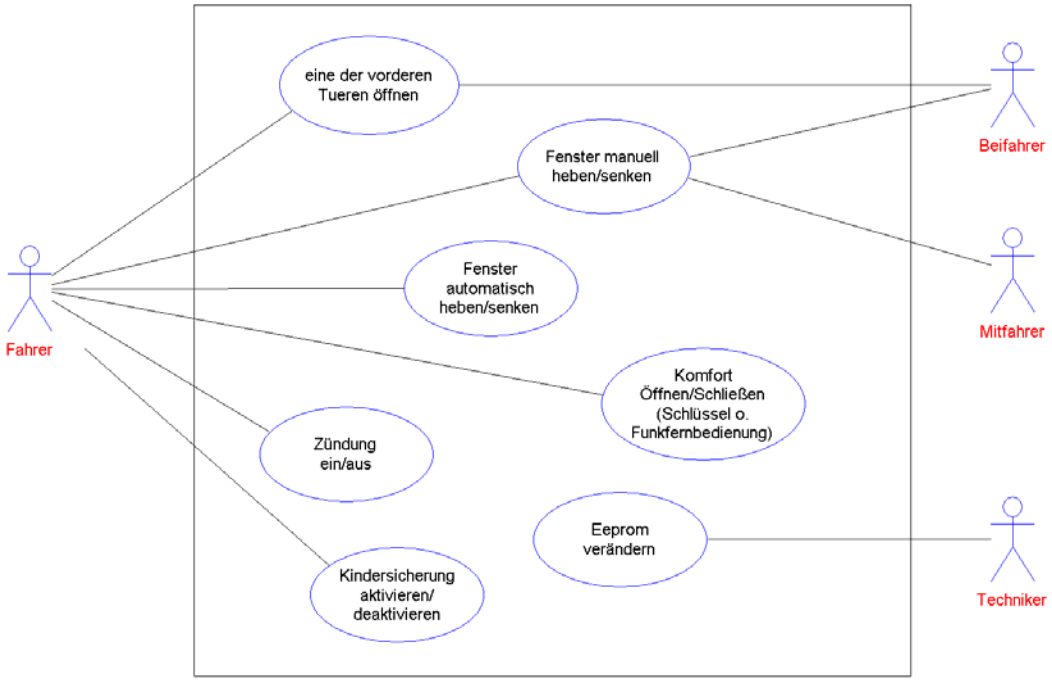
# Analyse: Ziel: Spezifikation der Funktionen

- Identifizierung von
  - Systemfunktionen und Akteuren,
  - Datenabhängigkeiten,
  - Signalflüssen,
  - Schnittstellen
- Detaillierung der Systemfunktionen
- Modellierung der Systemstruktur
- Notation: UML: Anwendungsfall-, Sequenz-, Klassendiagramme
- Werkzeug: Artisan RtS, andere typische UML-Werkzeuge



# Analyse in UML

## Anwendungsfälle und Akteure am Beispiel Fensterheber



# Abhängigkeiten und Vernetzung

- Vernetzung der Funktionen
  - Funktionale Abhängigkeiten
  - Feature Interaction
  - Vernetzung nach außen
- Neue Funktionen durch Funktionskomposition

Konsequenzen für die Entwicklung

- Gesamtarchitektur
  - Funktionen/Use Cases
  - Komponentenzerlegung
  - Wirkungszusammenhänge
- Mehr Augenmerk auf die Integration



# Logische Modellierung der Anforderungen

- In der logischen Modellierung werden logische Abhängigkeiten zwischen Systemereignissen oder -zuständen erfasst.

  Diese Erfassung kann informell durch Text erfolgen.

  Im Prinzip können logische Abhängigkeiten aber auch formal mit Mitteln der mathematischen Logik erfasst und analysiert werden.

  Vorteile mathematischer Logik:
  Es entsteht eine Modellierung mit exakt festgelegten Begriffen

- der Vollständigkeit
- der Widerspruchsfreiheit
- der logischen Implikation (was aus den Festlegungen zwingend folgt).

# Lastenheft/ Pflichtenheft

- Lastenheft: Dokument, das alle Anforderungen an ein System aus Sicht des Auftraggebers für eine Systementwicklung festhält.
  - Das Lastenheft enthält die Zusammenstellung aller Anforderungen des Auftraggebers hinsichtlich Liefer- und Leistungsumfang (aus: /VDI-VDE 3694/). In ihm sind die fachlichen Basisanforderungen aus Anwendersicht einschließlich aller Randbedingungen beschrieben. Damit enthält es die für den Hersteller des Softwaresystems relevanten Informationen zur Aufgabenstellung. Diese müssen quantifizierbar und überprüfbar sein. Im Lastenheft wird definiert, WAS WOFÜR zu lösen ist und nicht, WIE die Leistungen zu erbringen sind.
- Pflichtenheft: Detaillierte Anforderungen und Vorgaben für den Entwickler eines Systems
  - Vollständige, eindeutige, testbare Beschreibung einer Anforderung an das System.
  - Vollständig bedeutet hier, dass alle Details zu der Anforderung zu definieren sind. Es sollten so wenig wie möglich Aspekte als selbstverständlich eingeschätzt werden. Es ist in der Praxis davon auszugehen, dass diese Aspekte dann gerade so realisiert werden, wie es der Projekt-Auftraggeber nicht wollte.
  - Eindeutig meint, dass die Anforderung mit möglichst einfachen Worten so zu definieren ist, dass keine Missverständnisse zu erwarten sind.
  - Testbar müssen alle Anforderungen sein, da eine nicht testbare Anforderung nicht korrekt geprüft und daher niemals vom Auftraggeber abgenommen werden kann.



# Ausführbare Lastenhefte

- Sind die Modelle in einer festgelegten Form erfasst, die es erlaubt, Modelle auszuführen, so können sie zur Simulation und formalen Überprüfung verwendet werden
- Typischerweise werden Systeme dazu durch ein System verteilter Zustandsmaschinen dargestellt.

# Rapid Prototyping und Simulation

- Bestimme Techniken der Modellierung führen auf Beschreibungen, die ausführbar sind!
- Beispiel: Zustandsübergangsmodelle!
- Aus solchen Modellen können wir Code erzeugen oder die Modelle direkt interpretieren (ausführen, simulieren)!
- Dieses Vorgehen erlaubt folgende methodische Optionen:
  - ◇ Validierung der Spezifikation durch Durchspielen von Beispielsabläufen.
  - ◇ Generierung von Testfällen



# Entwurf: Zerlegung eines Systems in Komponenten

- Ausgehend von einer Anforderungsdefinition wird in einem Entwurf das System in Teilsysteme aufgespalten, die miteinander interagieren und damit die geforderte Funktionalität erbringen.
- Damit sind im Entwurf folgende Schritte zu erbringen:
  - ◇ Aufteilung eines Systems in Teilkomponenten
  - ◇ Festlegung der Kommunikationsbeziehung zwischen den Teilkomponenten
  - ◇ Festlegung der Rollen und des Verhaltens der Teilkomponenten (Schnittstellenspezifikation der Teilkomponenten)

# Architektur

- Eine Architektur definiert die Struktur (Aufteilung) eines Systems in Teilkomponenten.
- Bestandteile einer Architekturbeschreibung:
  - Komponenten
  - Rollen der Komponenten
  - Interaktion zwischen den Komponenten
  - Begründung der Architektur
- Die Architektur bestimmt maßgeblich die Qualität eines Systems in Hinblick auf Qualitätssicherung, Integration und Wartbarkeit



# Integration

- Die Architekturbeschreibung ist die Grundlage für die Systemintegration.
- Unter Systemintegration versteht man das Zusammenführen (Komponieren) eines Systems aus vorgefertigten Teilen und die Überprüfung, ob die Teile funktionsgerecht zusammenwirken.

Prinzip der Modularität (bei Komposition)

Bei der Konstruktion eines Systems aus Komponenten,

ergibt sich aus

der Kommunikationsbeziehungen zwischen den Komponenten und der Schnittstellen der Komponenten

das Schnittstellenverhalten des Gesamtsystems durch die Abstraktion des Verhaltens und Zusammenwirkens der Teilkomponenten

# Implementierung

- In einer Implementierung werden die in der Architektur und der Komponentenzerlegung vorgegebenen Module/Klassen durch Programme in einer Programmiersprache (C, C++, Java, Assembler etc.) realisiert.
- Ein Modul (oder in der Objektorientierung eine Klasse) ist eine Programmeinheit überschaubarer Größe, die Zustände und Funktionen kapselt und über eine spezifizierte Schnittstelle zur Verfügung stellt.
- Teile der Implementierung können durch geeignete Werkzeuge auch aus Modellbeschreibungen generiert werden.
- Bei der Implementierung empfiehlt es sich, bestimmte Codiervorschriften („Coding Standards" - Beispiel MISRA) vorzugeben, die bestimmte Qualitätsmerkmale des Codes begünstigen.



# Beteiligte Prozesse bei der Software/Systementwicklung

- Kernprozesse
  - Projektvorbereitung
  - Softwareentwicklung
  - Softwaremodifikation
- Unterstützende Prozesse
  - Projektmanagement
    - Rollenzuordnung
    - Dokumentation
    - Konfigurationsmanagement
    - Qualitätssicherung
  - Absicherungsprozess
    - Testprozess
    - Reviews
    - Problemlösungsprozess

# Projektvorbereitung Softwareentwicklung

- Eingangsinformationen beschaffen
  - Spezifikation der Einsatzumgebung und Schnittstellen des gesamten Steuergeräts
  - Spezifikation der Funktionalität Steuergeräts
  - Sicherheitsanforderungen festlegen (Gefahrenanalyse und Risikoeinstufung)
  - Nichtfunktionale Anforderungen
- Art der Softwarebereitstellung bestimmen
  - Neuentwicklung
  - Modifikation gegebener Software
  - Wiederverwendung gegebener Software
- Zusammenarbeit Lieferanten
- Festlegung der Vorgehensweise (Tailoring V-Modell)



# Softwareentwicklung

- Initiierung: Ziele festlegen
- Allgemeine Anforderungen Steuergerät
  - Steuergeräte Anforderungsspezifikation
  - Steuergerätearchitekturentwurf
- Software
  - Anforderungen
  - Architektur
  - Modulentwurf
  - Codierung/Debugging
  - Modultest
  - Integrationstest
  - Abnahmetest
- Steuergeräteintegration
  - Steuergeräteintegrationstest
  - Steuergeräteabnahmetest

# Qualitätsmerkmale Software

- Angemessenheit der Anforderungen: Anforderungen entsprechen den Zielen und Nutzerbedürfnissen
- Korrektheit: Implementierung entspricht den Spezifikationsvorgaben
- Zuverlässigkeit: Durchschnittliche Zeit bis Fehler auftritt
- Wartbarkeit: Strukturiertheit, Einfachheit Verständlichkeit
  - Änderbarkeit
  - Fehlerdiagnose
  - Erweiterbarkeit



# Qualitätssicherung

- Aktiv: Maßnahmen die Entwicklungsprodukte hoher Qualität sichern

  Beispiel:
  - Wohldefinierter Entwicklungsprozess
  - Standards für Entwicklungsprodukte
  - Werkzeugunterstützung

- Passiv: Maßnamen, die Entwicklungsprodukte auf Qualitätseigenschaften untersuchen

  Beispiele:
  - Test
  - Inspektion
  - Review
  - Abnahme

# Inspektion

- In einer Inspektion wird ein Entwicklungsprodukt einer eingehenden Begutachtung unterworfen, die mit einer schriftlichen Stellungnahme zur Qualität endet.
- Die Inspektion arbeitet mit bestimmten vorgegebenen Untersuchungsmethoden (Beispiel: Codeinspektion als Structured Walk Through auf Basis von Spezifikationen und Zusicherungen)



# Review

- Ein Review besteht in einer Sitzung aus Gutachtern (Reviewern), Vertretern des Entwicklerteams und Moderator sowie Protokollant.
- Ziel eines Reviews ist die Durchsprache und Bewertung der Qualität eines Entwicklungsproduktes.
- Ziel ist nicht ein Finden der Ursache oder Schuldigen für Probleme oder von Ansätze für ihre Lösung.

# Test

- Testen ist die heute wichtigste aber auch kostspieligste Methode der passiven Qualitätssicherung.

- Ein Test besteht
  - In der Auswahl eines Testfalls oder mehrer Testfälle.
  - Ein Testfall besteht in Festelegung einer Eingabe(folge) an ein System oder Teilsystem und in der Festlegung der nach Spezifikation geforderten korrekten Ausgabe(nfolge).
  - Das System oder Teilsystem (Programm) wird mit der festgelegten Eingabe(folge) zur Ausführung gebracht und die Ausgabe(nfolge) mit der korrekten Ausgabe(nfolge) verglichen.
  - Der Test war erfolgreich, wenn die durch die Ausführung generierten Ausgabe(nfolge) mit der erwarteten übereinstimmt.



# Grundsätzliches zum Testen

- Testen kann niemals die Korrektheit eines Programms beweisen, sondern nur seine Inkorrektheit (E.W. Dijkstra)
- Testen ist sehr aufwändig.
- Testen lässt sich automatisieren (wesentliche Aufwandsreduktion und Erhöhung der Effektivität durch höhere Anzahl der Testfälle)
- Schon bei der Spezifikation von Systemeigenschaften ist zu überlegen, wie diese zu testen sind.
- Es ist ein wesentlicher Unterschied zwischen probeweisen Programmläufen in der Entwicklung zur Fehlerbeseitigung (Debugging) und dem Testen nach Abschluss der eigentlichen Entwicklung.

# Grundsätzliches zum Testen

Aspekte zum Test

- Black Box Test: Testfälle werden aus der Außensicht heraus bestimmt
- Glass Box Test (White Box Test): Testfälle werden aus der Innensicht heraus bestimmt
- Modultest: Test der Module/Klassen
- Komponententest: Schnittstellentest der Komponenten
- Integrationstest: Test des Zusammenspiels der Komponenten
- Regressionstest: Wiederholung von Tests nach Änderungen/Fehlerkorrekturen
- Systemtest: Test des Gesamtsystems
- Belastungstests: Test unter hoher Systemlast



# Projektorganisation und Management

Neben den technischen Aufgaben der Projekt-durchführung ist eine angemessene Projekt-organisation und ein versiertes Projektmanagement für den Projekterfolg unverzichtbar.

# Aufgaben Projektorganisation

- Struktur des Unternehmens und der Entwicklungs-abteilung
  - Strukturierung der Fachzuständigkeit und der Softwarekompetenz
  - Wahl der Partnerfirmen
- Definition der Rollen und Zuständigkeiten
- Teambildung
- Kompetenzbildung und -erhaltung



# Aufgaben Projektmanagement

- Projektinitiierung
  - Teamformation
  - Rollenzuordnung
- Projektplanung
  - Kostenplanung
  - Arbeitsplanung
- Projektüberwachung und -steuerung (Controlling)
  - Anforderungs(Änderungs)management
  - Fortschrittskontrolle
  - Qualitätskontrolle
  - Erkennen und Beseitigung von Problemen
  - Dokumentation der Aufwände
  - Änderungsmanagement
  - Sicherung der Ergebnisse
    - Versionsmanagement
    - Konfigurationsmanagement
  - Zulieferermanagement
- Projektabrechnung, Sicherung der Erkenntnisse

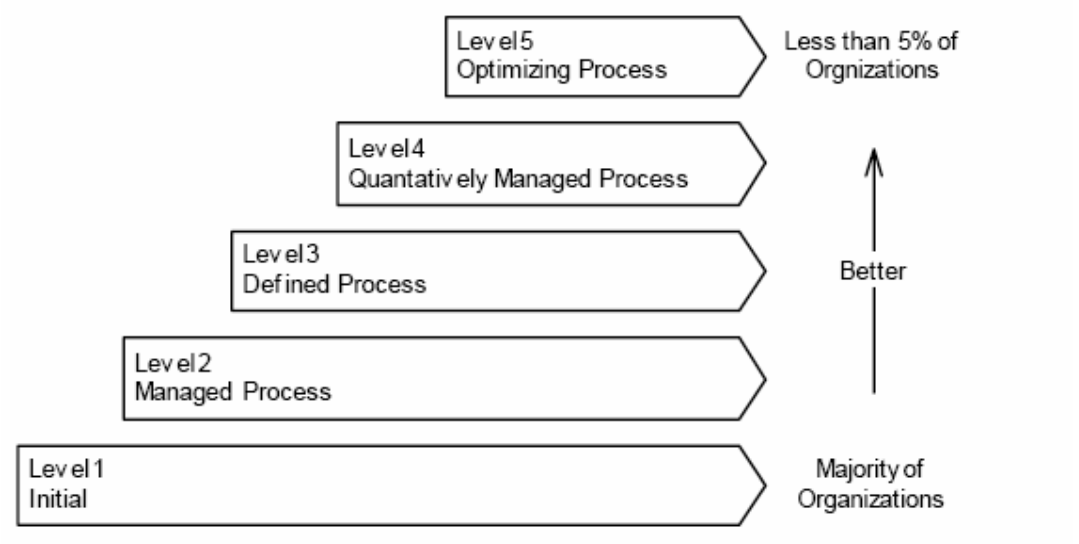
# Projektübergreifende Aufgaben

- Werkzeuge auswählen und bereitstellen
- Softwarebibliotheken/Plattformen auswählen und bereitstellen
- Kontinuierliche Verbesserung der Prozesse
- Schulung und Qualifikation



# Qualitätsstandards

- CMMI
- Spice



# CMMI - Überblick

| Maturity Level 1 | Maturity Level 2 | Maturity Level 3 | | Maturity Level 4 | Maturity Level 5 |
|---|---|---|---|---|---|
| No process area associated with the maturity level 1 | Requirements Management<br>Project Planning<br>Project Monitoring and Control<br>Supplier Agreement Management<br>Measurements and Analysis<br>Process and Product Quality Assurance<br>Configuration Management | Requirements Development<br>Technical Solution<br>Verification<br>Validation<br>Product Integration<br>Organizational Process Focus<br>Organizational Process Definition<br>Organizational Training | Integrated Project Management for IPPD<br>Integrated Supplier Management<br>Risk Management<br>Decision Analysis and Resolution<br>Integrated Teaming<br>Organizational Environment | Quantitative Project Management<br>Organizational Process Performance | Causal Analysis and Resolution<br>Organizational Innovation and Deployment |