

Martin Eigner

## Kurzfassung

Wissenschaftler aus diversen Disziplinen haben seit vielen Jahren Methoden und Vorgehensmodelle vorgeschlagen, um den Produktentwicklungsprozess (PEP) zu unterstützen. Diese sind in den meisten Fällen als prozessorientierte Richtlinien gedacht: Verschiedene Phasen der PEPs werden als Best Practices definiert, die einmalig oder zyklisch durchlaufen werden sollen. Zusätzlich werden auch die Entwicklungsergebnisse aus jeder Phase in den Vorgehensmodellen vorgeschrieben. Dieses Kapitel verschafft einen Überblick über disziplinspezifische, sowie für die Entwicklung multidisziplinärer Produkte relevante disziplinübergreifende Methoden und Vorgehensmodelle.

## Lernziele

In diesem Kapitel soll der Leser tiefer in die Disziplin-spezifischen und die Disziplin-übergreifenden entwurfstechnischen Methoden und Vorgehensmodelle eingeführt werden. Zu den ersten gehören die Methoden und Vorgehensmodelle der Mechanik, der Elektrotechnik und Elektronik sowie der Softwareentwicklung. Zu den zweiten gehören die Methoden und Vorgehensmodelle der Mechatronik und des Systems Engineering. Die Ähnlichkeiten und Unterschiede der verschiedenen Ansätze werden erarbeitet. Der Leser soll die Notwendigkeit interdisziplinärer Methoden und Vorgehensmodelle zur Entwicklung mechatronischer und cybertronischer Produkte verstehen.

---

M. Eigner (✉)

Lehrstuhl für Virtuelle Produktentwicklung,

Fachbereich Maschinenbau und Verfahrenstechnik, Technische Universität Kaiserslautern,

Gottlieb-Daimler-Straße 44, 67663 Kaiserslautern, Deutschland

E-Mail: [eigner@mv.uni-kl.de](mailto:eigner@mv.uni-kl.de)

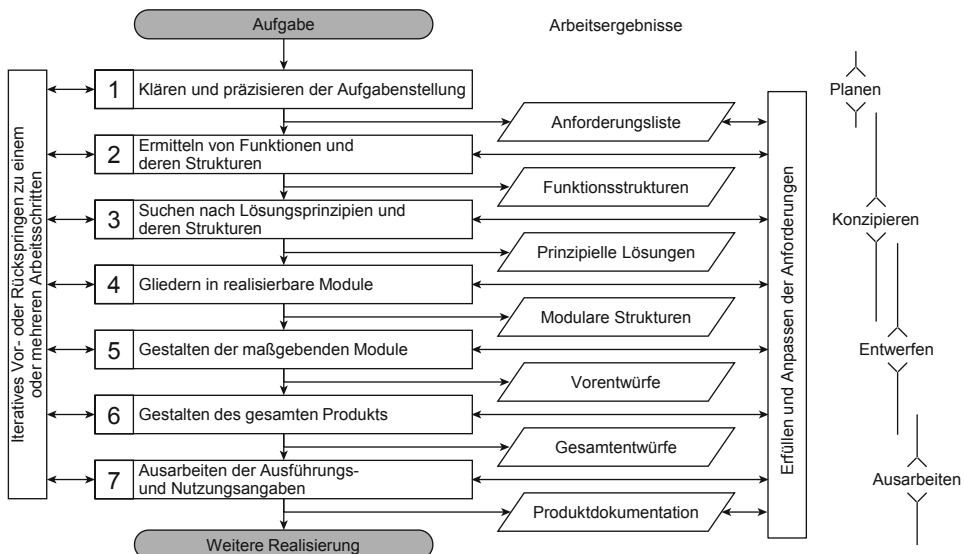
## 2.1 Vorgehensmodelle der Entwicklung mechanischer Produkte

Nahezu alle für die Mechanik etablierten Vorgehensmodelle (Andreasen [3], Pugh [50], Cross [11], Ehrlenspiel [17], Pahl/Beitz [48], French [22], Eder/Hosnedl [16], Malmqvist [45]) gehen von einem PEP aus, der aus vier Hauptphasen besteht.

- Anforderungs-/Aufgabenklärung, Planen
- Konzipieren
- Entwerfen und
- Ausarbeiten, Detaillieren

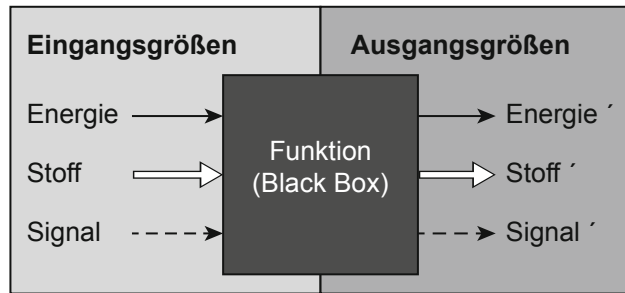
Sie definieren in der Konzeptphase einheitlich die Funktion und deren Umsetzung durch Prinziplösung als die wesentliche Elemente dieser Phase. Stellvertretend sei hier die VDI Richtlinie 2221 [57] dargestellt (Abb. 2.1), an der einige Konstruktionswissenschaftler mitgearbeitet haben und die zur damaligen Zeit so etwas wie eine gemeinsame Basis für den PEP für mechanische Produkte zumindest für den deutschen Sprachraum wurde.

Die Anforderungsliste ist das Ergebnis des ersten Arbeitsschrittes *Klären und Präzisieren der Aufgabenstellung*. Sie ist identisch mit der ersten der vier grundlegenden Konstruktionsphasen *Planen*. Sie bildet gleichzeitig das Dokument zur Produktspezifikation, sowie das Maß für den Grad der Aufgabenerfüllung für die Entwicklungs- und Konstruktionsabteilung. Zusätzlich enthält die Anforderungsliste Hinweise auf wichtige Einflüsse, Absichten oder solche zur Durchführung. Auf dieser Ebene hat sich eine eigene Disziplin gebil-



**Abb. 2.1** Phasen des PEP nach VDI 2221 [57]

**Abb. 2.2** Funktionsdefinition  
nach Pahl/Beitz [47]



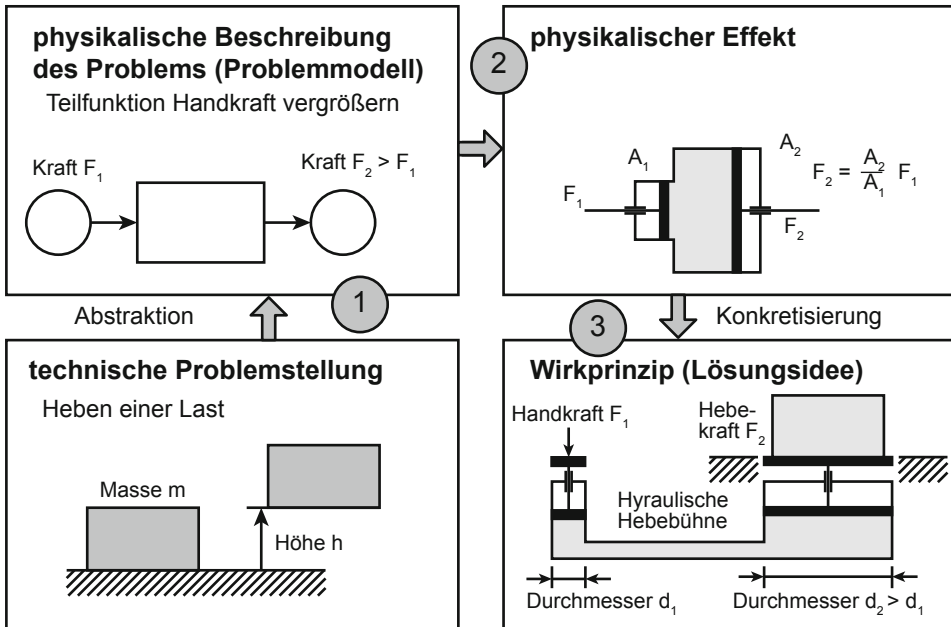
det: Requirements Engineering. Typische Aufgaben des Requirements Engineering sind die Identifizierung von Interessengruppen, Verständnis der Kundenbedürfnisse und der Identifikation, Analyse, Verfolgung und Validierung von Anforderungen.

In der zweiten Konstruktionsphase *Konzipieren* beginnt die Suche nach geeigneten *Funktionsstrukturen* und zugeordneten *prinzipiellen Lösungen*. Diese enthalten für die Funktionserfüllung erforderliche physikalische Effekte sowie die Wirkprinzipien mit geometrischen und stofflichen Merkmalen. Die Lösungsansätze der einzelnen Wirkprinzipien werden als Prinzipskizze dargestellt und bilden zusammengesetzt die Wirkstruktur. Abb. 2.2 zeigt die sehr moderne Definition der Funktion. Neben den Eingangsgrößen Energie und Stoff wird bereits das Signal definiert und damit eine Tür zur Mechatronik geöffnet. In Abb. 2.3 wird die Abbildung von Problemstellung auf Funktion und auf das Wirkprinzip dargestellt.

Die Konzeptphase wird als besonders wichtig anerkannt. Sie eröffnet den größten Spielraum für signifikante Verbesserungen [22]. Entscheidungen, die in dieser Phase getroffen wurden, haben Auswirkungen auf alle nachfolgenden Planungsphasen. Die große Bandbreite der Gestaltungsmöglichkeiten stellt eine Herausforderung für Produktentwickler dar, denn hier müssen Ingenieurwissenschaften, praktisches Wissen, Produktionsverfahren und kommerzielle Aspekte in den PEP eingehen. In dieser Phase wird ein wesentlicher Teil der Gesamtkosten definiert. Das Thema Kostenverursachung versus Kostenverantwortung wird von vielen Autoren angesprochen und ist in der VDI Richtlinie 2235 [56] dargestellt (Abb. 2.4).

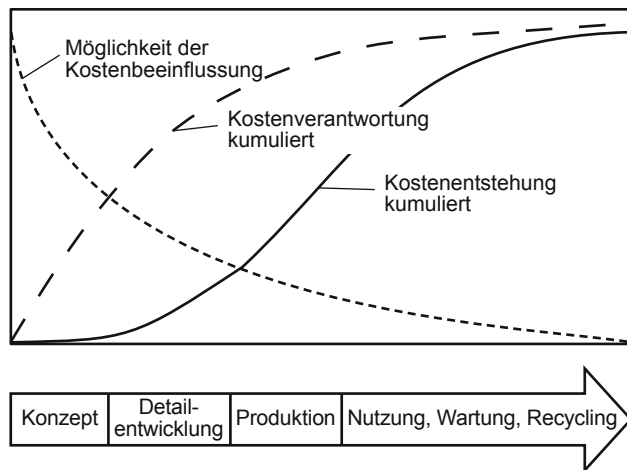
In der *Entwurfsphase* werden die prinzipiellen Lösungen zunächst in realisierbare Module gegliedert. Hieraus resultiert eine *modulare Struktur*, welche zusätzlich zur Funktions- und Wirkstruktur, schon eine Gliederung der Lösung des Problems inklusive deren Gruppen, Elementen und Verknüpfungen beinhaltet.

Das Erstellen dieser Strukturen wird nötig, um besonders bei komplexen Konstruktionen, die aufwändigen Gestaltungs- und Konstruktionsarbeiten besser aufteilen zu können und eine effektivere Arbeitsweise zu ermöglichen. So werden auch Konstruktionsschwerpunkte bestimmt; man unterscheidet nach Arbeitsmodulen mit arbeitstechnisch-pragmatischen Schwerpunkten, Montagemodulen zur montagegerechten Produktgestaltung, Wartungsmodulen für einen instandhaltungsfreundlichen Produktaufbau, Recyclingmodulen sowie



**Abb. 2.3** Abbildung Problem – Funktion – physikalischer Effekt – Wirkprinzip (nach [49])

**Abb. 2.4** Verhältnis Kostenverantwortung zu Kostenverursachung (in Anlehnung an [7, 18, 21])



Variationsmodule, die eine Baukastenstruktur ermöglichen. So verzweigt sich in diesem Stadium der Produktentwicklung die Konstruktion in eine parallele Bearbeitung von Baugruppen und Einzelteilen. Damit ist die Grundlage für die Erstellung von *Vorentwürfen* gelegt.

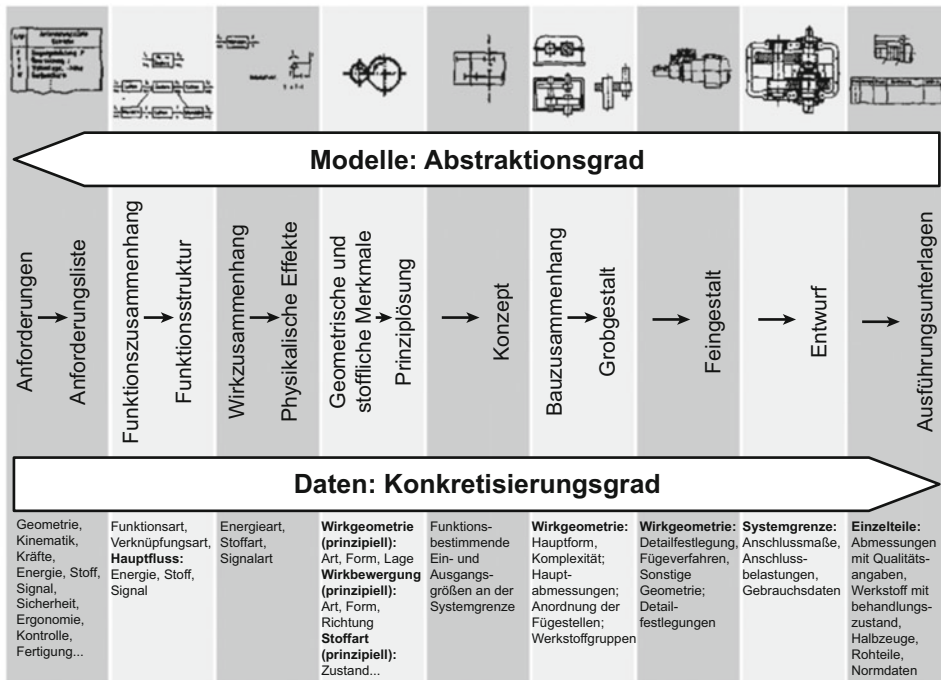
Dieser Teil des Konstruierens beinhaltet die gestalterische Festlegung der Lösung [48]. Der Grundstein für diese Phase wurde mit dem Festlegen der Wirkstruktur und der Prinziplösung erarbeitet, sodass im Folgenden die für das Produkt, die Produkt- sowie die Systemoptimierung erforderlichen Module realisiert werden können. Hierzu gehört vor allem die Wahl der Werkstoffe, der Fertigungsverfahren, die Festlegung der Abmessungen, der Untersuchung der räumlichen Verträglichkeit und dem Erarbeiten der Lösungen für die Nebenfunktionen. Hierbei steht immer die technologische und wirtschaftliche Betrachtung der Lösungen im Vordergrund, weshalb der Abschluss dieser Phase auch mit einer technisch-wirtschaftlichen Bewertung abgeschlossen wird. [48, 57] Der Entwurfsprozess ist gekennzeichnet durch ein iteratives Vorgehen. So wird zunächst die Struktur nur grob erarbeitet, so dass ein Erkennen und Auswählen des Gestaltungsoptimums möglich ist, um letztendlich die Vorentwürfe zu erhalten. [VDI93]. Im nächsten Arbeitsschritt „Gestalten des gesamten Produkts“ wird ein *Gesamtentwurf* erzeugt (Abb. 2.2), der eine Beurteilung der Funktionen, der Haltbarkeit, der Fertigungs- und Montagemöglichkeit, der Gebrauchseigenschaften sowie der Kostendeckung erlaubt.

Nach der Erstellung des Gesamtentwurfs schließt sich das *Ausarbeiten* der Ausführungs- und Nutzungsangaben an. Hierbei werden die Ausführungs- und Nutzungsangaben von den zuständigen Entwicklungs- und Konstruktionsbereichen erarbeitet. In der ersten Phase der Ausarbeitung, dem Detaillieren und Festlegen, werden für das Produkt die endgültigen Vorschriften für Form, Abmessungen, Oberflächenbeschaffenheiten und Werkstoffe festgelegt sowie eine Überprüfung der Herstellungs-, Gebrauchsmöglichkeiten und Kosten durchgeführt, um verbindliche Unterlagen für die stoffliche Verwirklichung und Nutzung zu erhalten. Danach folgt der Schritt des Zusammenfassens. Der Schwerpunkt hierbei liegt in der Erarbeitung von Fertigungsunterlagen, besonders die der Einzelteil- oder Werkstattzeichnungen, Gruppenzeichnungen und, soweit erforderlich, der Gesamtzeichnung und Stückliste. Dieser Teil der Ausarbeitungsphase wird stark von modernen 3D-CAD- sowie PLM-Lösungen unterstützt. Des Weiteren sind bei der Vervollständigung das Erstellen von Montage-, Transport und Prüfvorschriften zur Qualitätssicherung, sowie für den späteren Gebrauch die Betriebs-, Wartungs- und Instandsetzungsanleitungen, zu berücksichtigen. Zusätzlich müssen für die Arbeitsvorbereitung die Unterlagen zur Fertigungsplanung und Fertigungssteuerung fertiggestellt werden. [48]. Am Ende schließt sich die Prüfung an. Dieser Schritt ist besonders für die Fertigung wichtig und beinhaltet die Kontrolle der Einhaltung von Normen, der fertigungsgerechten Bemaßung, erforderlicher Fertigungsangaben und Beschaffungsgesichtspunkten.

Eine vollständige Übersicht des Produktentwicklungsprozesses ist in Abb. 2.5 dargestellt [21].

Die verschiedenen Ansätze den Konstruktionsprozess zu beschreiben, weisen trotz vieler Gemeinsamkeiten einige Unterschiede auf:

- Ab welcher Phase werden produktionsrelevante Fakten beachtet?
- Ab welcher Phase werden kostenrelevante Fakten berücksichtigt?
- Es wurden neben überwiegend sequentiellen Prozessen mit mehr oder weniger Iterationen, auch zyklische und spiralförmige Prozesse entwickelt.



**Abb. 2.5** Übersicht über den Produktentwicklungsprozess (nach [21])

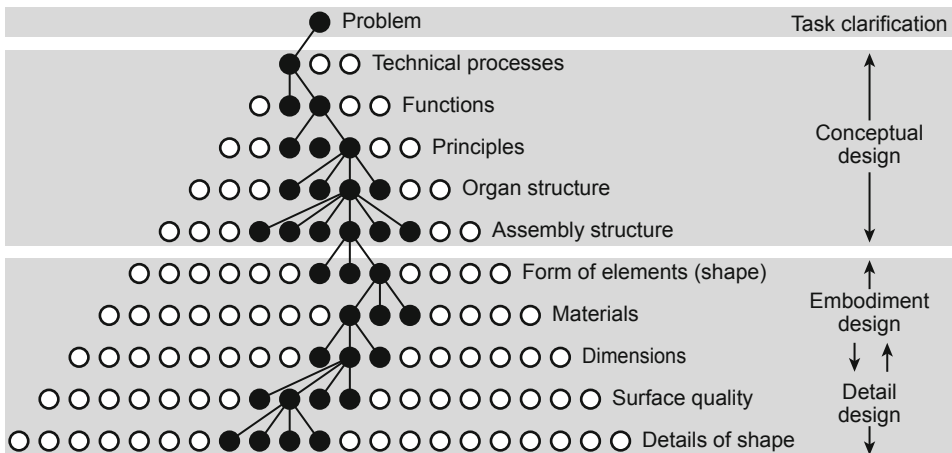
Wynn und Clarkson [66] unterscheiden darüber hinaus noch, ob der Ansatz

- Lösungs- oder Problemorientiert,
- Abstrakt, prozedural oder analytisch

ist.

Ein anderer interessanter Aspekt, der gerade im Vergleich zu Konstruktionsmethoden in der Elektronik und der Softwareentwicklung interessant ist, bezieht sich auf die Verwendung des Artefaktes „Verhalten“. Es ist auffallend, dass nur ganz wenige Autoren aus der Sicht der mechanischen Produktentwicklung über Verhalten sprechen. Die Beziehungskette geht i.d. R von Anforderungen zu Funktionen und dann weiter über physikalische Effekte zu Wirkprinzipien.

Andreasen und Hein [4] sprechen von miteinander verbundenen Funktionen, Wirkprinzipien und „Organen“. Mit letzteren meint er die physikalischen Lösungen. Der Begriff Organ wird im übertragenen Sinne verwendet. Interessant ist, dass er die beiden Begriffe zusammenfasst zu Verhalten (engl.: Behavior) (Abb. 2.6). Hier handelt es sich bei der typischen methodischen Vorgehensweise der Mechanik um ein postskriptives Verhalten. Der Mechaniker gibt eine Funktion mit Parametern vor, beschreibt ihre physikalische/geometrische Ausprägung und kann dann über die mathematisch/physikalischen Gesetzmäßigkeiten das Verhalten errechnen, simulieren und/oder physisch testen. Er beschreibt also das Ver-



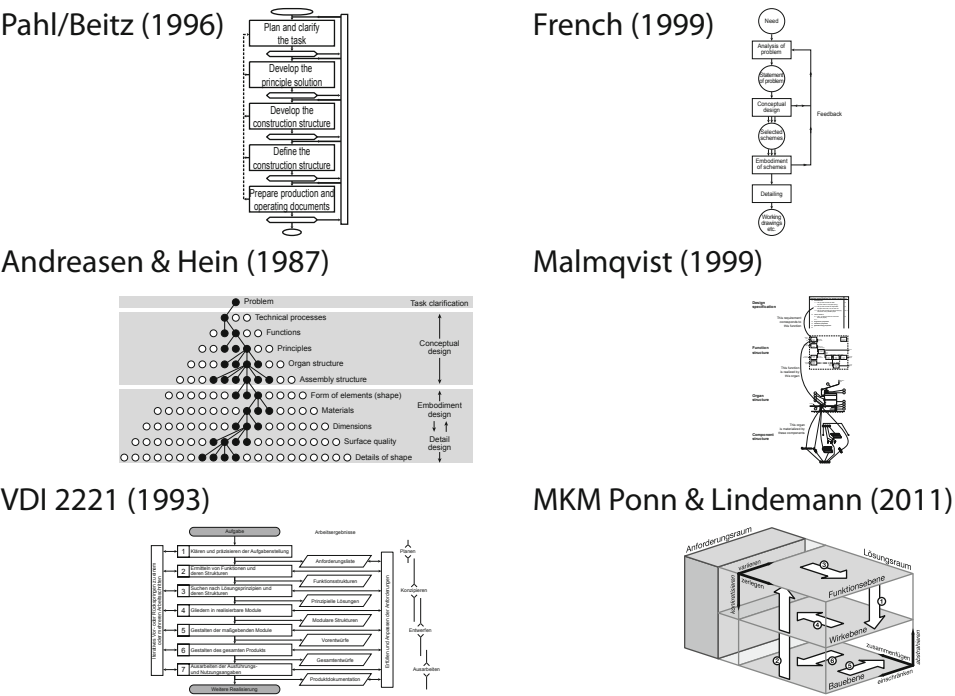
**Abb. 2.6** Entwurfsvorgehen von Andreasen & Hein [4], abgebildet auf die vier Standard Entwicklungsphasen

halten nicht, sondern ermittelt es am Entwicklungsergebnis. Der Elektroniker und der Informatiker definieren Verhalten präskriptiv als Teil ihrer Anforderungsspezifikation. Sie beschreiben also das gewünschte Verhalten. Dies ist in diesen Disziplinen einfacher, da es i.d. R durch Logiken und einfache Regeln beschrieben werden kann. Abbildung 2.7 zeigt zusammenfassend einen Überblick der verschiedenen Methoden und Vorgehensmodelle in der mechanischen Produktentwicklung.

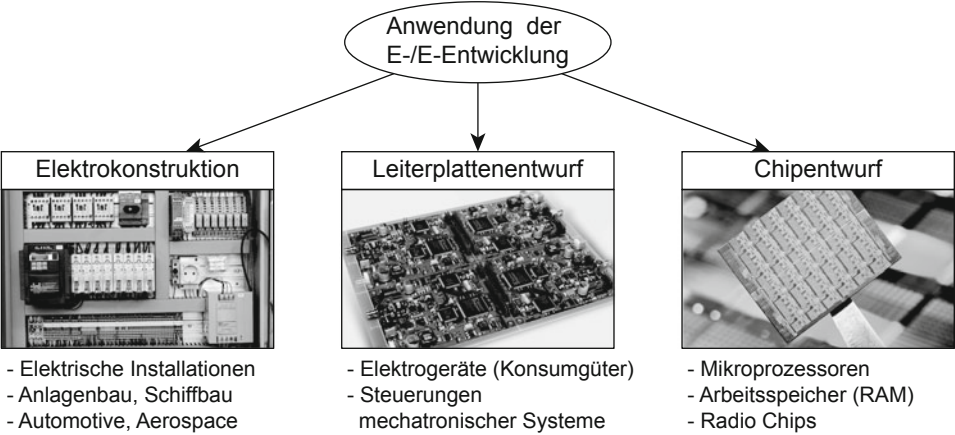
## 2.2 Vorgehensmodelle in der Elektrotechnik und Elektronik

Die Betrachtung der Konstruktionsmethoden in der Elektrotechnik und Elektronik (E/E) ergibt ein breiteres Bild als in der Mechanik. Das mag an vier Dingen liegen:

- Den sehr unterschiedlichen Anwendungen im Bereich der Elektrotechnik und Elektronik (Abb. 2.8).
- Der grundsätzlich verschiedenen Vorgehensweise des Konstrukteurs in der Elektrotechnik und Elektronik gegenüber der Mechanik, die eine Entwurfsebene des schematischen Entwurfs vor der geometrischen Ausgestaltung (Layout) beinhaltet. Diese schematische Ebene besitzt bereits simulierbare funktional-logische Elemente (Abb. 2.9).
- Dem rasanten Technologiewandel insbesondere beim digitalen Schaltungsentwurf. Komplexität und Integrationsdichte nehmen permanent zu. Dabei steigen die Anforderungen an höhere Packungsdichte, geringere Strukturgröße, Leistungsfähigkeit, geringerer Platzbedarf, hohe Taktrate und geringerer Leistungsverbrauch [39].
- Der Evolution der Automatisierungstechniken bezüglich logischem und physikalischen Entwurf sowie der Verifikation (Abb. 2.10) [61].



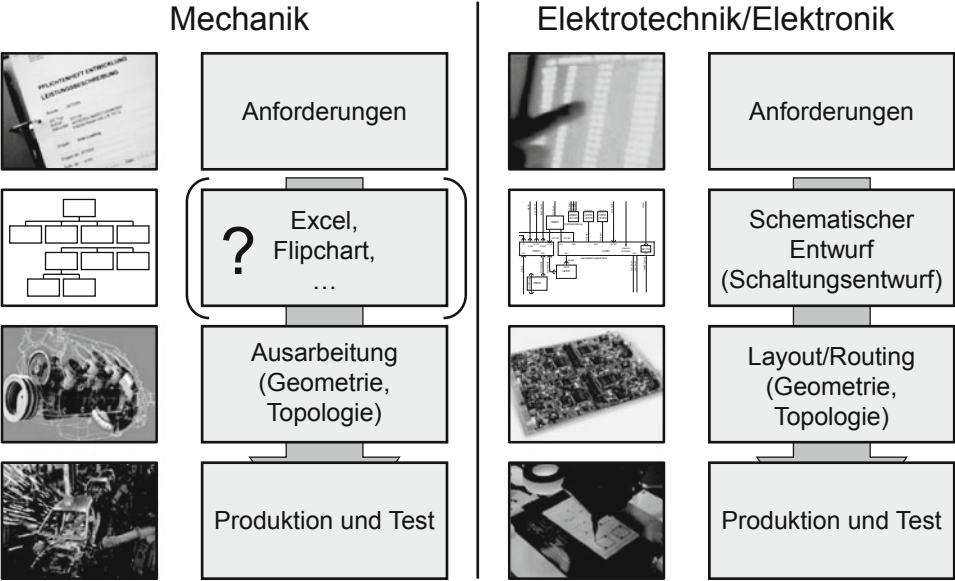
**Abb. 2.7** Überblick über Methoden und Vorgehensmodelle der Mechanikkonstruktion



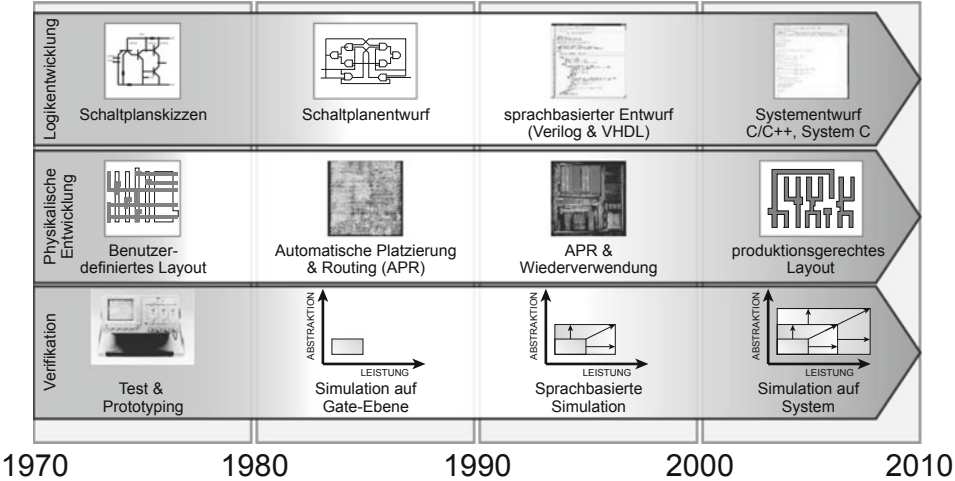
**Abb. 2.8** Anwendungsgebiete der Elektrokonstruktion und Elektronikentwicklung

Nach Kümmel [38] und Stephan [54] existieren jedoch in den verschiedenen Anwendungsgebieten unterschiedliche Entwurfsstrategien. So lehnt sich zum Beispiel die VDI/VDE Richtlinie 2422 [58] für den Entwurf von Geräten, die von Mikroelektronik gesteuert werden, stark an die VDI Richtlinie 2221 an. Der Schwerpunkt der Richtlinie liegt in der Beschreibung des konkreten Vorgehens bei der Geräteentwicklung. Dazu werden die Vor-



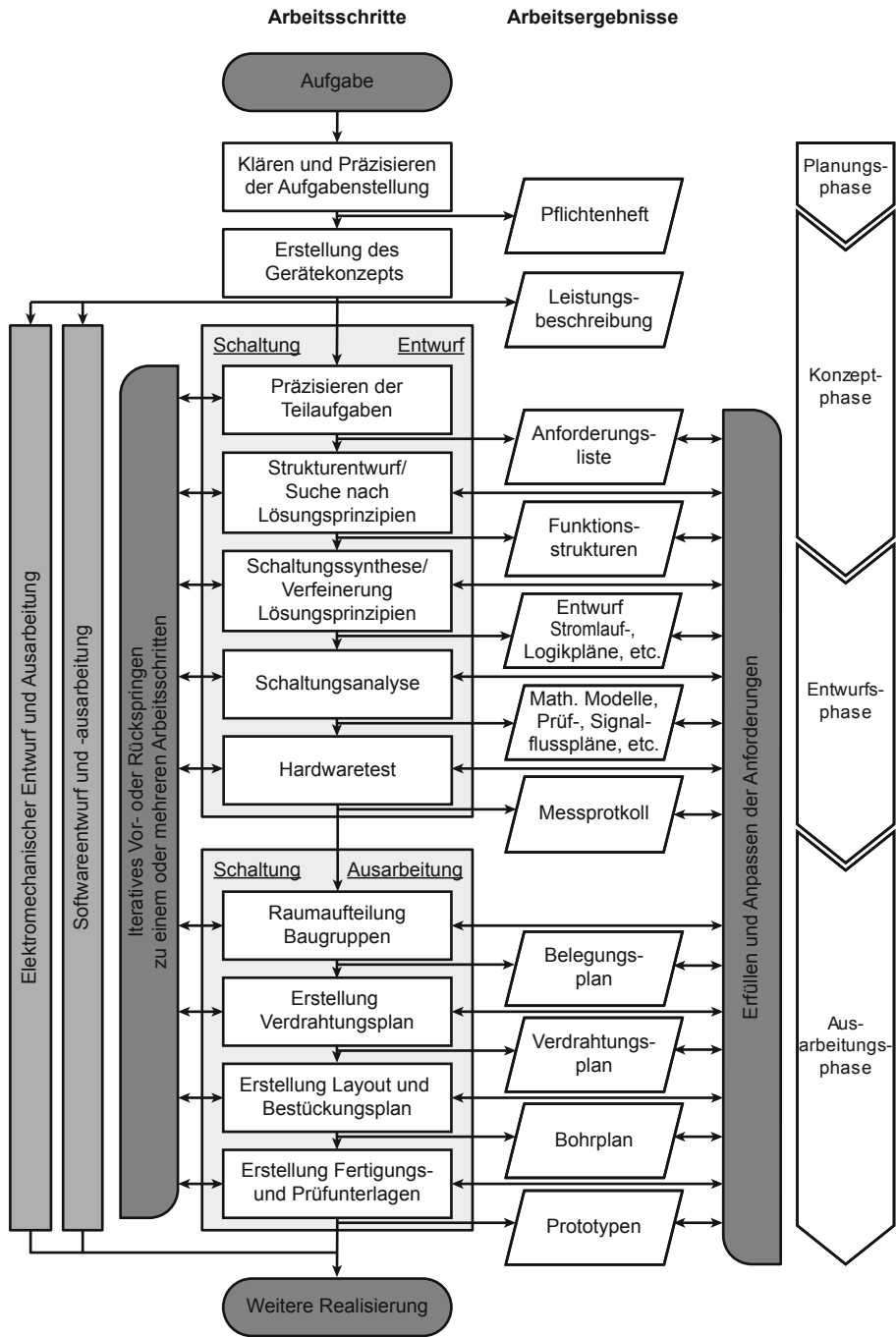


**Abb. 2.9** Vorgehensweise in der Mechanik und in der Elektrokonstruktion bzw. Elektronikentwicklung



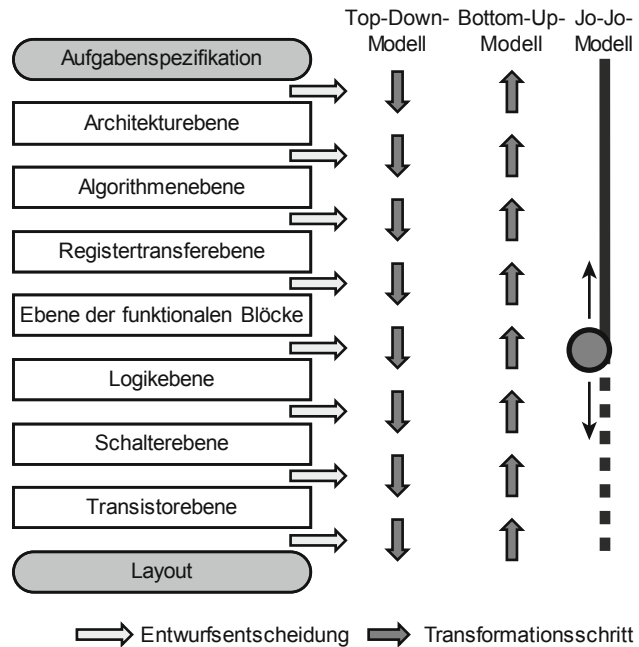
**Abb. 2.10** Evolution der Entwurfsautomatisierung der Elektronik (nach [61])

gehensschritte in den Bereichen Softwareentwicklung, Schaltungsentwicklung und bei der Entwicklung des elektromechanischen Geräteteils detailliert beschrieben. Hierzu zeigt die VDI 2422 Wege auf, wie sich die Entwicklungsaufgabe in zeitliche und fachliche Abschnitte gliedern lässt (Abb. 2.11). Durch die Beschreibung der Schnittstellen und der Definition von allgemeinen Begriffsinhalten wird ein zum Anwenden der verschiedenen Methoden notwendiges begriffliches Verständnis hergestellt.



**Abb. 2.11** Methodisches Vorgehen bei der Schaltungsentwicklung in Anlehnung an VDI 2422 [58] und Kümmel [38] (nach [54])

**Abb. 2.12** Vorgehensweisen beim Entwurf technologieunabhängiger Modelle (in Anlehnung an [51, 54])

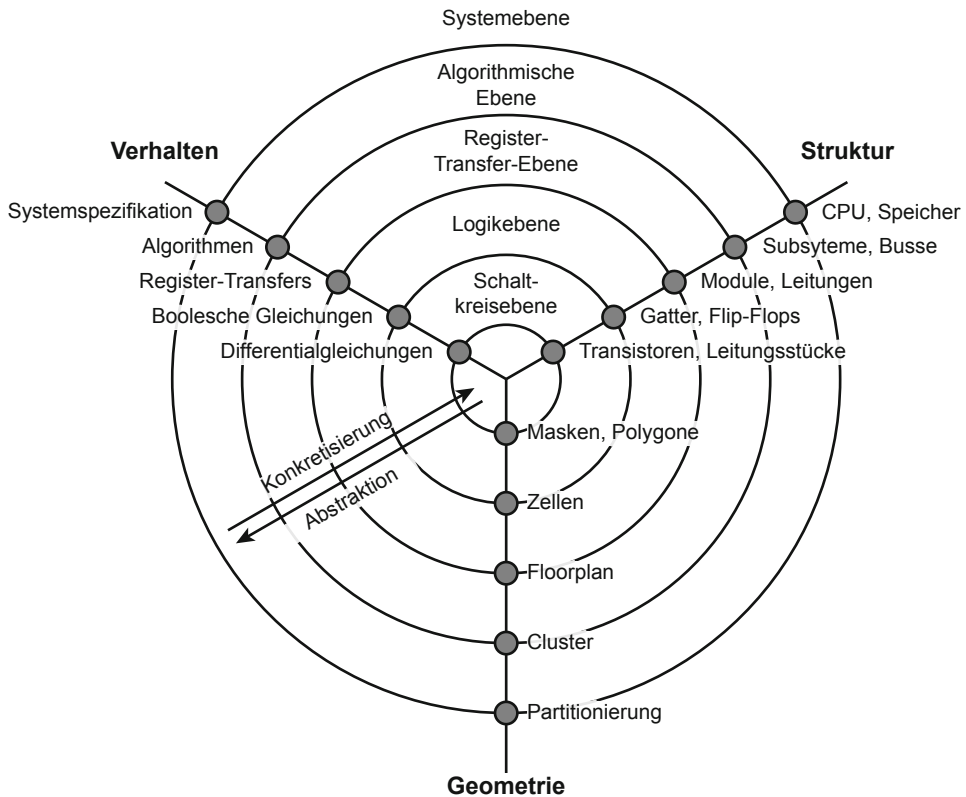


Im Anwendungsgebiet des digitalen Schaltungsentwurfs liegen viele Modelle des Entwurfsprozesses vor, die sich auf den klassischen Abstraktionsebenen wenig unterscheiden:

- Systemebene,
- Algorithmische Ebene,
- Register-Transferebene,
- Logikebene und
- Schaltkreisebene.

Ein wesentliches Klassifikationsmerkmal für die jeweiligen Entwurfsmodelle ist der Grad an Technologieunabhängigkeit. Zu diesen Modellen gehören z. B. der Top-Down und der Bottom-Up Entwurf [44, 51], sowie das aus beiden hervorgegangene Jo-Jo Modell [54] (Abb. 2.12).

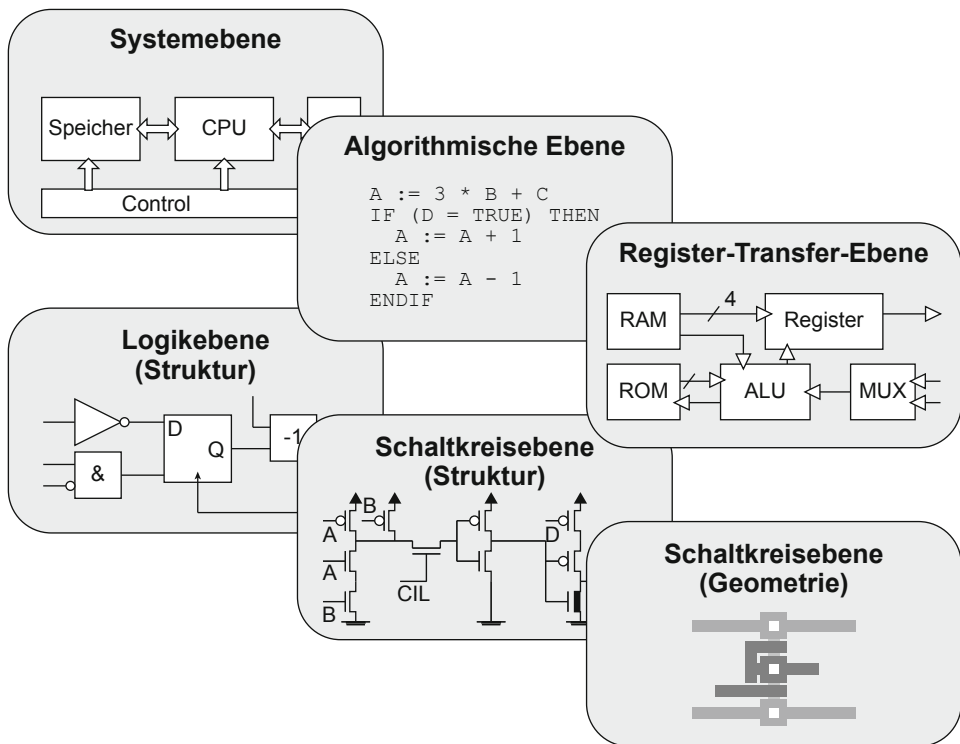
Zu den bekanntesten technologieunabhängigen Modellen zählt das *Y-Diagramm* (auch als *Gajski-Diagramm* bekannt). Es beschreibt die Sichtweisen im Hardwareentwurf, insbesondere bei der Entwicklung von integrierten Schaltkreisen. Im Jahr 1983 entwickelten Daniel D. Gajski und Robert Kuhn das Entwurfsmodell [25]. Im Jahr 1985 wurde es von Robert Walker und Donald Thomas [60] verfeinert (Abb. 2.13). Gajski unterscheidet bei diesem Modell für den Entwurf von Hardware drei verschiedene Domänen:



**Abb. 2.13** Das Y-Diagramm nach Gajski-Walker (in Anlehnung an [39])

- Verhalten,
- Struktur und
- Geometrie

Die Domänen sind als Achsen des Y dargestellt. Die verschiedenen Abstraktionsebenen werden durch konzentrische Kreise repräsentiert. Außen ist der Abstraktionsgrad am höchsten. Vereinfacht kann der Entwurf von integrierten Schaltkreisen als eine Reihe von Transformationen (Wechsel) der Sichtweise auf einem Abstraktionskreis und Detaillierung (Wechsel der Abstraktionsebenen auf einer Achse des Y-Modells) darstellen [39]. Beim Hardwareentwurf handelt es sich zumeist um ein Top-Down-Design, bei dem Verhalten, Struktur und Geometrie hinunter zu geringeren Abstraktionsebenen beschrieben werden. Generell kann der Entwickler sich nach dem Modell des Y-Diagramms eine der Sichtweisen aussuchen und zwischen den Sichtweisen und Abstraktionsebenen hin- und herspringen. Somit ist der Entwurfsprozess auf keine bestimmte Reihenfolge im

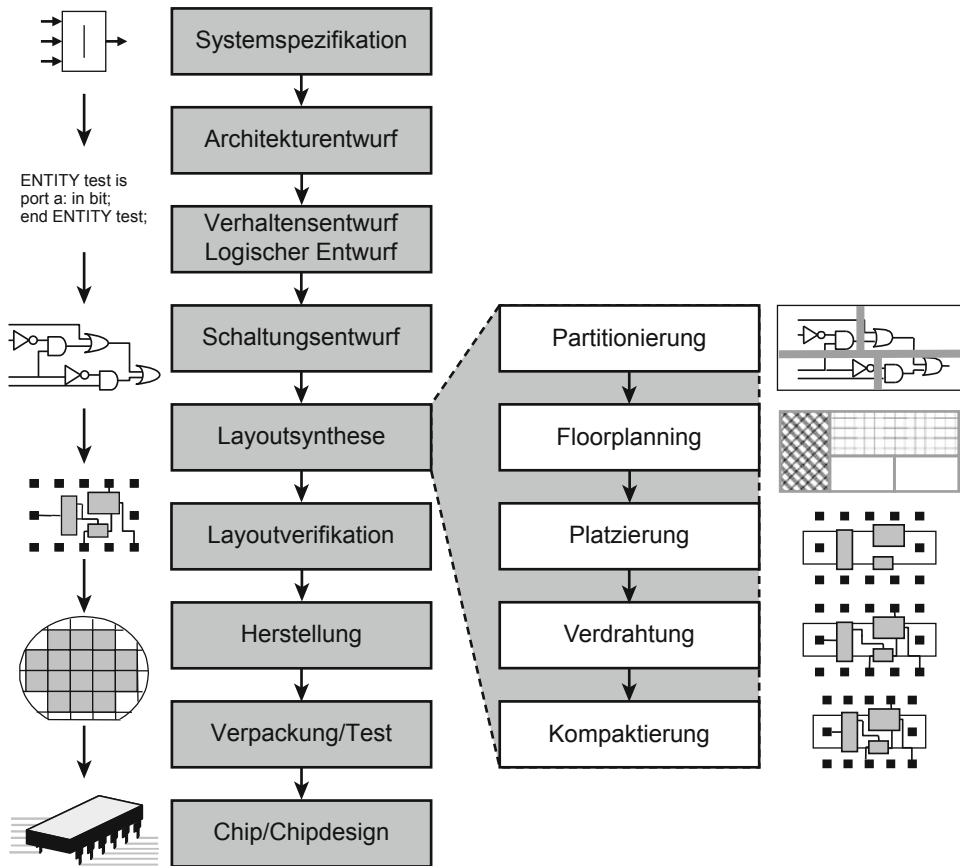


**Abb. 2.14** Ebenen beim Entwurf elektronischer Systeme (nach [39])

Y-Diagramm festgelegt [39]. Interessant ist die Existenz der Achse „Verhalten“, die typische für die Elektronikentwicklung ist.

Nach Lehmann, Wunder und Selz [39] gliedern sich die Entwurfsebenen wie folgt (Abb. 2.14):

- **Systemebene**  
Auf dieser Ebene werden die grundlegenden Kriterien eines elektronischen Systems beschrieben. Es enthält logische Blöcke wie Speicher, Prozessoren und Schnittstellen. Zeitverhalten, Signale und funktionales Verhalten sind nicht Bestandteil dieser Ebene.
- **Algorithmische Ebene**  
Die Beschreibung dieser Ebene besteht aus sequenziell und parallel ablaufenden Algorithmen und Logiken. Typische Beschreibungselemente sind Funktionen, Verhalten, Prozeduren, Prozesse und Kontrollstrukturen. Auf dieser Ebene kommunizieren die Blöcke bereits über Signale. Die Verhaltenssicht enthält Logiken, Algorithmen, Variablen und Operatoren.
- **Register-Transfer Ebene**  
Auf dieser Ebene werden die Eigenschaften einer Schaltung incl. zeitlichem Verhalten durch Operationen, z. B. Addition oder Subtraktion, und durch die Zuweisung von Daten zu Registern determiniert.



**Abb. 2.15** Wesentliche Schritte beim digitalen Schaltkreisentwurf (nach [40])

- **Logikebene**  
Diese Ebene beschreibt das elektronische System durch logische Verknüpfung und deren zeitliche Eigenschaften. Der Verlauf der Ausgangs- und Eingangssignale ergibt sich aus der Anwendung der Verknüpfung. Auf der Strukturachse bieten Bibliotheken die typischen Grundelemente (AND, OR, XOR, Flip-Flops) an.
- **Schaltkreisebene**  
Auf dieser Ebene werden aus struktureller Sicht elektronische Bauelemente (Transistoren, Widerstände, Kapazitäten) vernetzt und die einzelnen Module durch den tatsächlichen Aufbau und Verknüpfung der Bauelemente beschrieben. Die Verhaltenssicht nutzt überwiegend Differentialgleichungen zur Modellierung des Systemerhaltens.

Ein weiterer pragmatischer Ansatz für einen Entwurfsprozess integrierter Schaltungen, wurde von Lienig entwickelt [40] (Abb. 2.15).

Dieser geht von einer Einteilung des Entwurfs in sequenziell zu bearbeitende Schritten aus. Diese sind ähnlich dem Y-Modell in unterschiedliche Abstraktionsebenen gegliedert. Steht am Anfang noch das Gesamtsystem im Vordergrund werden mit jedem weiteren Entwurfsschritt zunehmend Entwurfsdetails relevant. Auf der letzten Stufe sind alle Schaltelemente mit ihren strukturellen, geometrischen und elektrischen Eigenschaften beschrieben. Im Folgenden werden die einzelnen Schritte beim Entwurf einer digitalen integrierten Schaltung kurz dargestellt [40]:

- **Systemspezifikation**  
Die Systemspezifikation entspricht der Anforderungsbeschreibung in den Entwurfssystemen der Mechanik. Wesentliche Entwicklungsziele wie Leistungsanforderung, Funktionalität, Abmessungen und Technologie werden festgelegt.
- **Architekturentwurf**  
Auf dieser Ebene wird die Grundarchitektur des Systems definiert. Dazu gehören Speicher, Prozessoren und Schnittstellen (↪ Systemebene beim Y-Modell)
- **Verhaltensentwurf/Logikentwurf**  
Die wesentlichen funktionalen Einheiten des Systems inklusive ihrer Verknüpfung werden vereinbart. Wesentlich ist dabei die Verhaltensbeschreibung. Bei der Weiterentwicklung zum Logikentwurf werden Steuersignale, Wortbreite und arithmetische sowie logische Operationen definiert. Unterschieden wird zwischen Daten- und Steuerfluss (↪ Algorithmische Ebene des V-Modells).
- **Schaltungsentwurf**  
Der Schaltungsentwurf benutzt die sogenannte RTL- (Register Transfer Level) Beschreibung und entspricht genau dieser Entwurfsebene beim Y-Modell. „High Level Synthesis Tools“ unterstützen heute eine weitgehende Automatisierung dieses Entwicklungsabschnitts.
- **Layoutsynthese**  
Auf dieser Ebene werden unter Nutzung von Bibliotheks- und Technologieinformationen die logische Schaltung (Netzliste) in ihre reale Geometrie überführt. „Dabei werden alle Schaltungselemente (Zellen/Gatter, Makrozellen, Transistoren usw.) in ihrem geometrischen Abbild (Form, Abmessung, Ebenenzuordnung) dargestellt und ihre räumliche Anordnung (Platzierung) sowie die konkreten Verbindungsstrukturen (Verdrahtung) zwischen ihnen ermittelt. Im Ergebnis liegt die Layoutdarstellung der Schaltung vor, die nach ihrer Verifikation ebenenspezifisch auf Masken übertragen wird und so die Herstellung der integrierten Schaltung ermöglicht“ [40]. Die Layoutsynthese bestimmt wesentlich die Leistungsfähigkeit des zu erstellenden Schaltkreises (Fläche, Zuverlässigkeit, Ausbeute des Herstellungsprozesses, usw.) Dies entspricht der Schaltkreisebene des Y-Modells). Aufgrund seiner Komplexität wird die Layoutsynthese in Unterabschnitte eingeteilt [40]:
  - Partitionierung (Aufteilung einer Schaltung in Teilschaltungen bzw. Schaltungsblöcke, die einzeln entworfen werden können).

- Floorplaning (Festlegung der Formen und der Anordnung der Schaltungsblöcke sowie der Belegungen der Außenanschlüsse).
- Platzierung (exakte Anordnung aller Zellen in einem Schaltungsblock).
- Globalverdrahtung (Zuordnung von Zellenverbindungen zu Verdrahtungsregionen).
- Verdrahtung der Stromversorgungs- und Massenetze.
- Verdrahtung der Taktnetze (Clock-Tree-Synthese).
- Feinverdrahtung der Signalnetze (exakte Zuordnung von Spuren innerhalb der bereits zugewiesenen Verdrahtungsregionen).
- Kompaktierung (Optimierung der Layoutfläche bzw. anderer Schaltungsparameter).
- Layoutverifikation  
Das Layout wird anschließend einer umfassenden Verifikation auf logische und elektrische Korrektheit (Electrical Rule Check ERC) sowie auf seine technologische Produzierbarkeit (Design Rule Check DRC) unterworfen.
- Herstellung  
Die Layoutinformationen werden an die Produktion i.d. R in Form von GDSII<sup>1</sup>-Daten transferiert. In der Produktion werden die lagenspezifischen Layoutinformationen in photolithografische Masken übertragen. Mit Hilfe dieser Masken werden auf dem Silizium Flächen definiert, wo Material auf- oder abgetragen werden soll. Auf einer Siliziumscheibe (Wafer) entstehen parallel eine Vielzahl von integrierten Schaltungen. Typische Waferdurchmesser liegen zwischen 200 mm und 300 mm. Die einzelnen „unverpackten“ Schaltungen (Dies) werden auf dem Wafer getestet und anschließend in die einzelnen Dies zersägt.
- Verpackung/Test  
Alle als „gut“ gekennzeichneten Dies werden einzeln verpackt, wobei verschiedene Verpackungsarten existieren:
  - Dual-In-Line Packages (DIP)
  - Pin Grid Arrays (PGA) und
  - Ball Grid Array (BGA)
- Die jeweilige Verpackungsart hängt von der späteren Anwendung ab.
- Die integrierten Schaltungen werden entweder vor oder nach der Verpackung auf Einhaltung der Entwurfsspezifikationen getestet.

Bei der Entwicklung von integrierten Schaltungen werden zur formalen Modellierung der Entwurfsebenen sogenannte Hardware Description Languages (HDL) verwendet. Sie dienen zur Spezifikation und Dokumentation der Architektur und des Verhaltens von integrierten Schaltkreisen (Abb. 2.16).

Die erste und häufig verwendete HDL Sprache (High-Speed Integrated Circuit Hardware Description Languages) ist VHDL. Sie wurde durch das US Department of Defense im Jahr 1983 initiiert und ist seit 1987 als IEEE-Standard festgelegt. Seitdem hat sich neben

---

<sup>1</sup> GDSII (ursprüngl. Graphical Design Station II oder Graphic Data System II) bezeichnet das De-facto-Standard-Datenformat für Layoutdaten von integrierten Schaltkreisen im EDA-Bereich.



```
architecture behavioral_seq of halfadder is
begin
    process (sum_a, sum_b)
    begin
        if (sum_a = '1' and sum_b = '1') then
            sum <= '0'; carry <= '1';
        else
            if (sum_a = '1' or sum_b = '1') then
                sum <= '1'; carry <= '0';
            else
                sum <= '0'; carry <= '0';
            end if;
        end if;
    end process;
end behavioral_seq;
```

**Abb. 2.16** Deklarieren der Architektur eines Halbaddierers in VHDL (nach [39])

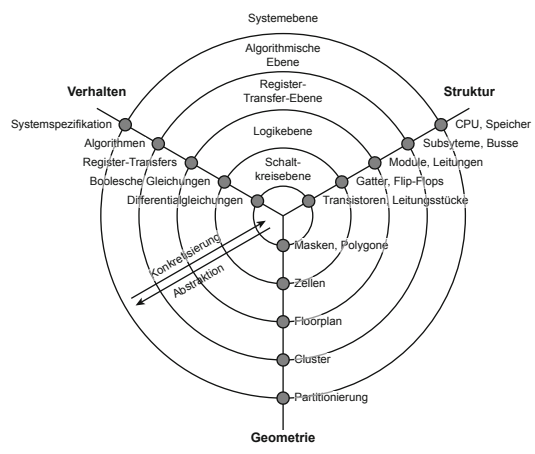
VHDL Verilog zur Standard-Hardware-Beschreibungssprache entwickelt. Der Sprachstandard (Syntax und Semantik) wird regelmäßig überarbeitet (IEEE 1076 93, IEEE 1076 02, IEEE 1076 08) und seit 2004 ist VHDL als IEC Standard (IEC 61691-1-1 04) weltweit festgelegt. SystemC ist eine Modellierungs- und Simulationssprache insbesondere für die Entwicklung von komplexen elektronischen Systemen, die sowohl Hardware- als auch Softwarekomponenten enthalten [65]. Während VHDL und Verilog-HDL zur reinen Hardwarebeschreibung eingesetzt wird, ist SystemC auch zur Modellierung auf höhere Abstraktionsebenen eingesetzt. Damit werden Simulationen um den Faktor 100 bis 1000 schneller. Weitere wesentliche Vorteile von SystemC sind daneben die freie Verfügbarkeit als Open Source und die Verwandtschaft zu der Programmiersprache C++ . SystemC ist keine eigenständige Sprache, sondern eine Klassen-Bibliothek für C++ [65].

Abbildung 2.17 zeigt nochmals eine Zusammenfassung von wesentlichen Methoden und Vorgehensmodellen der Elektrotechnik und Elektronik.

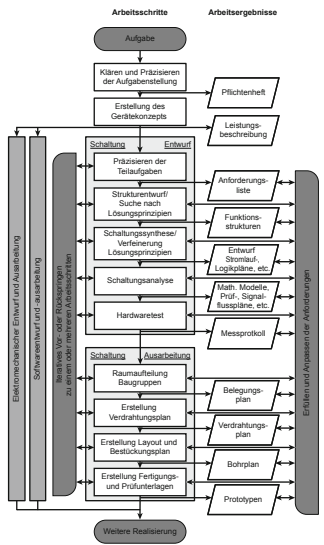
## 2.3 Vorgehensmodelle in der Softwareentwicklung

Aufgrund des hohen Aufwandes zur Erstellung und Wartung komplexer Software erfolgt die Entwicklung anhand strukturierter Vorgehens- bzw. Phasen- und Prozessmodelle. Diese Modelle unterteilen den Entwicklungsprozess in überschaubare, zeitlich und inhaltlich begrenzte Phasen. Die Software wird somit Schritt für Schritt fertiggestellt. Die Phasen sind während des ganzen Entwicklungsprozesses eng miteinander verzahnt. In der Praxis werden auch Verfahren eingesetzt, welche die Mehrstufigkeit von Systemanalyse, Systemdesign/Konzept und anschließender Implementierung und Testen aufgeben. Man nennt diese Methoden Agile Softwareentwicklung. Helmut Balzert beschreibt das Gebiet der Softwareentwicklung (engl.: Software Engineering) als [5]:

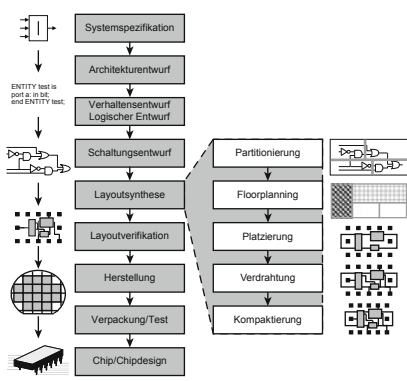
Y-Chart Gajski and Kuhn (1983)



VDI/VDE (1993)



Layoutsynthese Liening (2006)



Rauscher (1996)

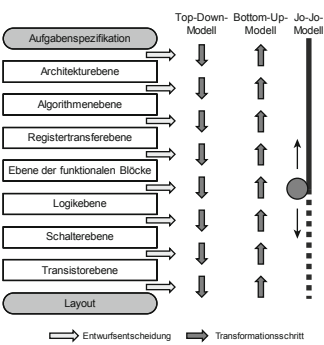
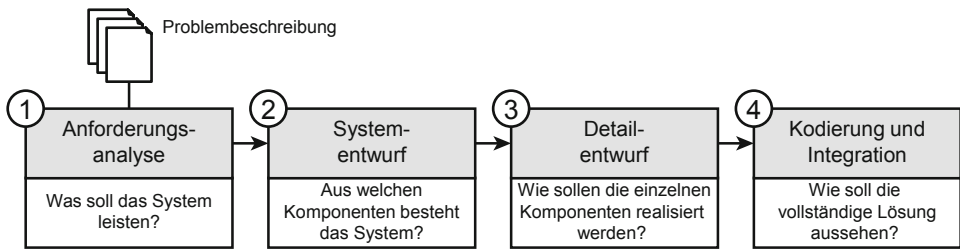


Abb. 2.17 Überblick über Methoden und Vorgehensmodelle der Elektrik/Elektronik

**Definition „Softwareentwicklung“ (engl.: Software Engineering):**  
„Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen“ [5].

Das *Phasenmodell* in Abb. 2.18 beschreibt die typischen Tätigkeiten in der Softwareentwicklung, gegliedert in vier Phasen, sowie die darin zu Grunde liegende Fragestellung. Das Phasenmodell gibt die Softwareentwicklung aus einer Tätigkeitssicht wieder [19].



**Abb. 2.18** Phasenmodell der Softwareentwicklung (nach [19])

Ziel der *Anforderungsanalyse* ist eine möglichst vollständige Erhebung der Anforderungen an das zu entwickelnde System. Dies kann mittels Diagrammen und/oder textueller Beschreibungen erfolgen. Im Wesentlichen geht es darum zu verstehen, was das System leisten soll. Hierbei werden Eingaben und Ausgaben, Benutzerschnittstellen und Funktionalitäten festgelegt. Ergebnisse aus der Anforderungsanalyse dienen als juristische Grundlage für spätere Abnahmetests

Ziel des Software-orientierten Systementwurfs ist es, einen abstrakten Plan der Lösung für das Problem zu beschreiben. Auf Basis der Anforderungsanalyse wird festgehalten, aus welchen Komponenten das System besteht. Die typischen Tätigkeiten des Systementwurfs sind:

- Es werden Komponenten bestimmt.
- Es wird definiert und zugeordnet, welche Aufgaben den Komponenten zukommen.
- Die Schnittstellen bzw. die Interaktionen zwischen den Komponenten werden definiert.
- Zu den Komponenten übergeordnete Datenflüsse und Steuerungen werden festgelegt.

Missverständnisse sind Hauptursachen für Fehler. Die im Systementwurf erarbeiteten Dokumente sollen möglichst präzise sein, um solche zu minimieren.

Ziel des *Detailentwurfes* ist es, den Plan für die Problemlösung aus dem Systementwurf weiter zu konkretisieren, d. h. zu beantworten, wie die einzelnen abstrakten Komponenten und deren Interaktion realisiert werden. Zusammenfassend bedeutet das folgende Tätigkeiten:

- Anforderungen und Komponenten werden verfeinert.
- Datenflüsse und Zustände werden auf Komponentenebene definiert.

Ziel der letzten Phase ist die *Kodierung und Integration* (Implementierung) der vollständigen Lösung. Dies erfordert folgende Tätigkeiten:

- Strukturen (Daten, Algorithmen) aus dem Detailentwurf werden auf die Konstrukte einer Programmiersprache abgebildet.

Prozessmodell	Primäres Ziel	Antreibendes Moment	Benutzerbeteiligung	Charakteristika
<i>Wasserfall</i>	minimales Management	Dokumente	gering	sequentiell, volle Breite
<i>V-Modell</i>	maximale Qualität	Dokumente	gering	sequentiell, volle Breite, V&V
<i>Prototypen</i>	Risikominimierung	Code	hoch	nur Teilsysteme (horizontal oder vertikal)
<i>Evolutionär</i>	minimale Zeit	Code	mittel	sofort: nur Kernsystem
<i>Inkrementell</i>	minimale Zeit & Risiko	Code	mittel	volle Definition, dann zunächst nur Kernsystem
<i>Nebenläufig</i>	minimale Zeit	Zeit	hoch	volle Breite, nebenläufig

**Abb. 2.19** Übersicht über prozessorientierte Softwareentwicklungsmethoden (nach [41])

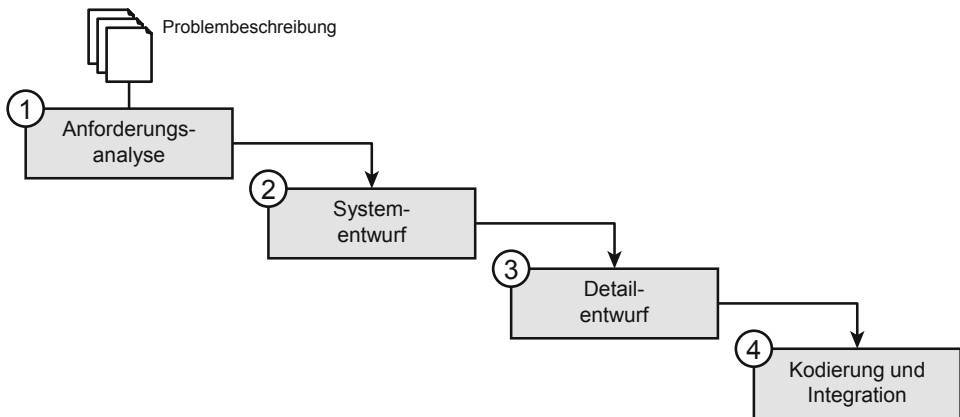
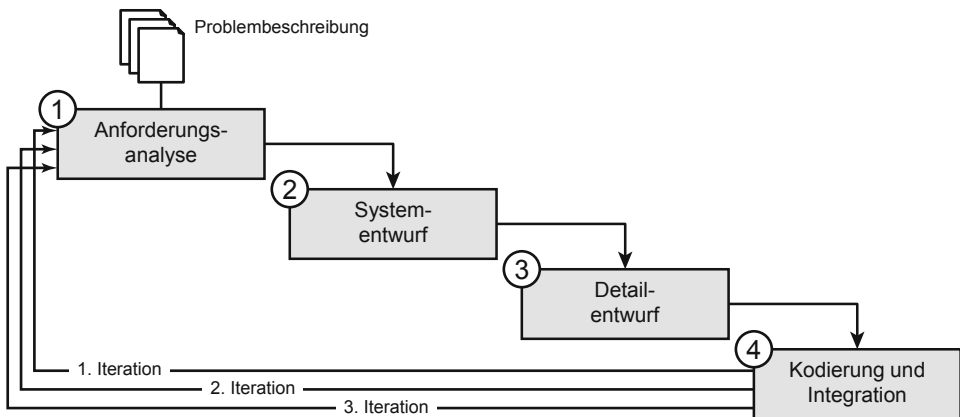
- Anforderungen an die Performanz, Robustheit und Ergonomie werden gegebenenfalls berücksichtigt.

Im Folgenden werden einige prozessorientierte Methoden der Softwareentwicklung vorgestellt. *Prozessmodelle* definieren den Ablauf bei der Softwareentwicklung [19]. Sie legen die Aktivitäten mit ihren Ergebnissen und deren Reihenfolge fest. Die Aktivitäten sind mit Rollen verknüpft, die die notwendigen Erfahrungen, Kenntnisse und Fähigkeiten der zugeordneten Mitarbeiter beschreiben [41]. Eine Übersicht dieser Methoden zeigt Abb. 2.19.

Im Wasserfallmodell geht man davon aus, dass eine Phase nach Abschluss nicht erneut durchlaufen wird. Das heißt eine Phase soll nicht beginnen, bevor die vorangegangene Phase vollständig abgeschlossen ist. Für die Anforderungen bedeutet das insbesondere, dass diese nach Beenden der Anforderungsanalyse in Phase 1 fehlerfrei spezifiziert sein müssen. Abb. 2.20 verbildlicht diesen Prozess.

In der Realität liegt einem Entwicklungsprozess inklusive der Anforderungsdefinition eine Dynamik zu Grunde, so dass sich das Wasserfallmodell als nicht geeignete Methode erweist. Das iterative Prototypenmodell geht vergleichsweise davon aus, dass ein Softwaresystem inkrementell entwickelt wird. Das Modell sieht Änderungen während der Entwicklung vor, d. h. die Entwickler legen sich nicht sofort auf die vollständigen Anforderungen fest. Dies erlaubt einen Grad an Flexibilität beim Entwurf und der Implementierung, was jedoch einen erhöhten Bedarf an Wartung und Verwaltung verursachen kann (Abb. 2.21).

Als weiteres Softwareentwicklungsmodell soll das Spiralmodell nach Boehm aus dem Jahr 1979 vorgestellt werden. Es hat das Ziel Entwicklungsrisiken im Softwareentwicklungsprozess darzustellen und zu minimieren. Der Softwareentwicklungsprozess startet im

**Abb. 2.20** Wasserfallmodell**Abb. 2.21** Iteratives Prototypenmodell

Zentrum der Spirale (Abb. 2.22). Je länger ein Projekt läuft, umso weiter bewegt sich das Entwicklerteam vom Inneren der Spirale nach außen. Sukzessive wächst die Software und damit auch die Kosten und die Komplexität. Für jede Entwicklungsrunde werden zu Beginn ein Budget, Ziele, Rahmenbedingungen und mögliche Alternativen definiert. Danach werden diese Alternativen evaluiert. Zur Unterstützung bei der Evaluierung können Prototypen erstellt werden. Nach der Evaluierung in der ersten Runde können schon risikoarme Softwareanforderungen und diese validiert werden. Die nächsten Entwicklungsschritte werden dann im Rahmen einer Anforderungs- und Lebenszyklusplanung definiert. Das Prinzip wiederholt sich bis am Ende ein fertiges Produkt entstanden ist.

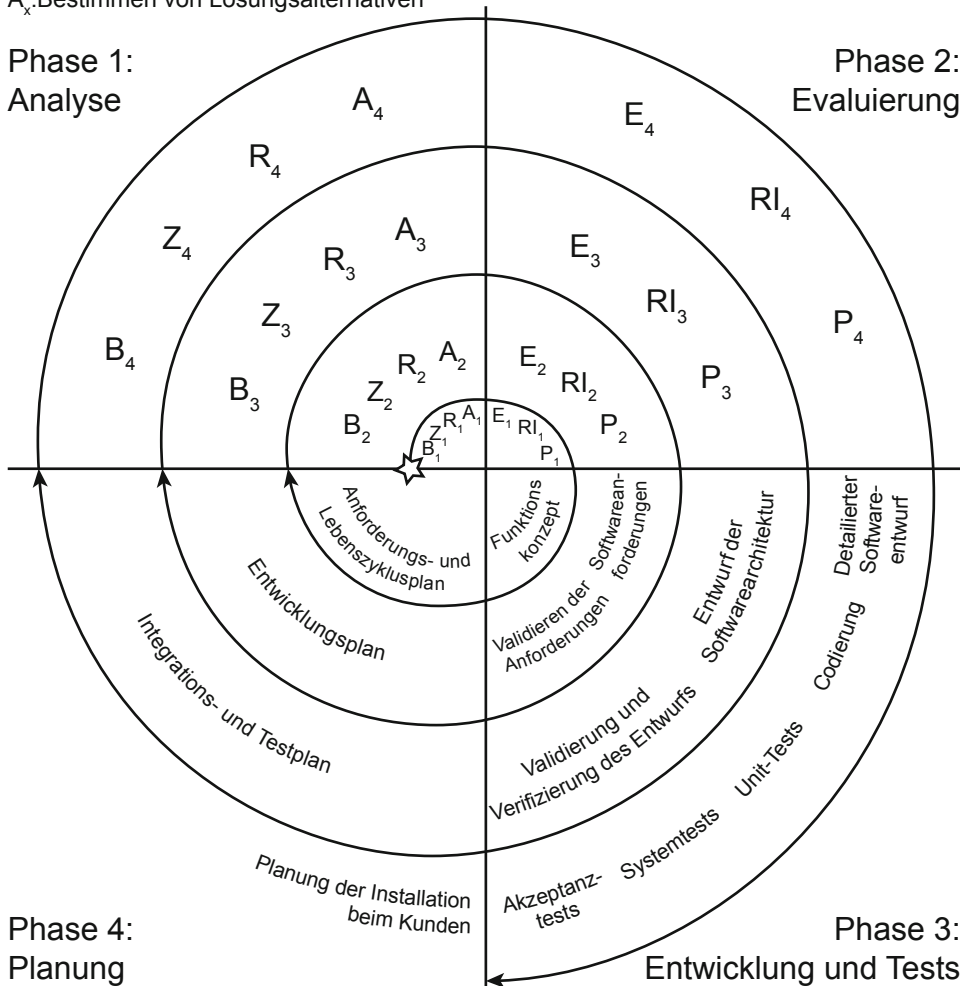
Die in den vorangehenden Kapiteln erläuterte Richtlinie VDI-2206 beschreibt eine V-förmige Sicht auf die Produktentwicklung, die ihren Ursprung in der Informatik findet [9] und in der Literatur in unterschiedlichsten Ausprägungen existiert. Der Software-Ingenieur

Legende:

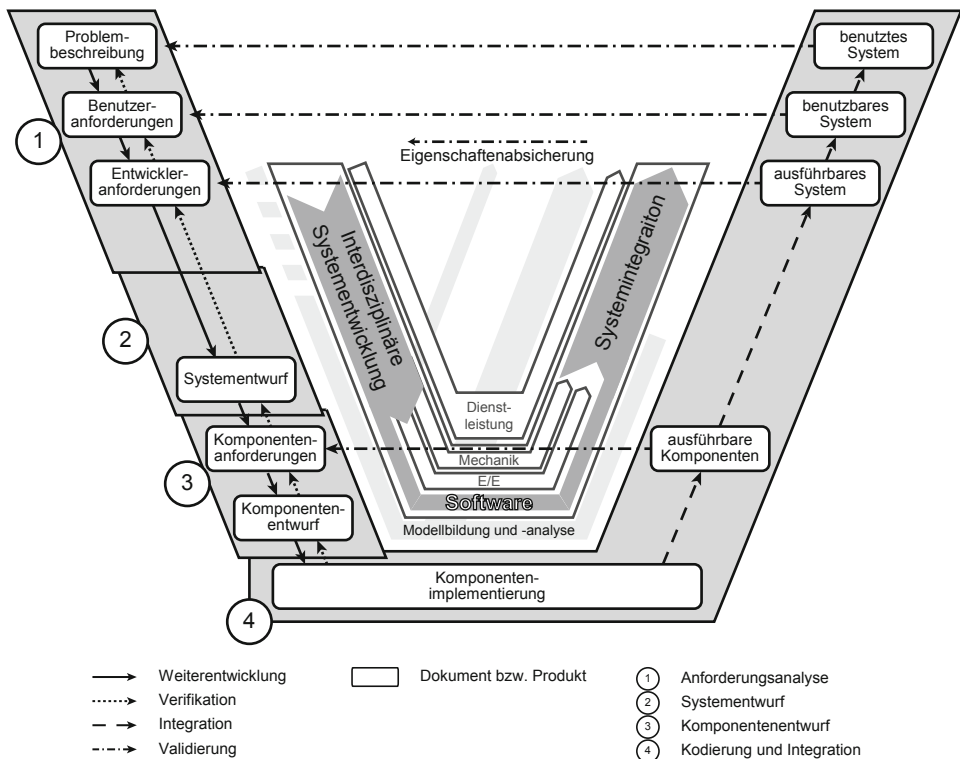
X: Runde

 $B_x$ :Planung des Budgets $Z_x$ :Festlegen der Ziele $R_x$ :Festlegen der Rahmenbedingungen $A_x$ :Bestimmen von LösungsalternativenLegende:

X: Runde

 $E_x$ :Evaluierung der Lösungsalternativen $RI_x$ Erkennen und bewerten von Risiken $P_x$ :Prototyp erstellenPhase 1:  
AnalysePhase 2:  
Evaluierung**Abb. 2.22** Das Spiralmodell nach Boehm [9]

spricht oftmals beim sogenannten V-Modell von einem *Dokumenten- und Produktmodell*, um darzustellen welche Dokumente und Teil-Produkte (oft nur Produkte genannt) bei der Softwareentwicklung anfallen, und wie diese miteinander in Verbindung stehen. Abb. 2.23 fasst dies mitsamt den Phasen der Softwareentwicklung zusammen.

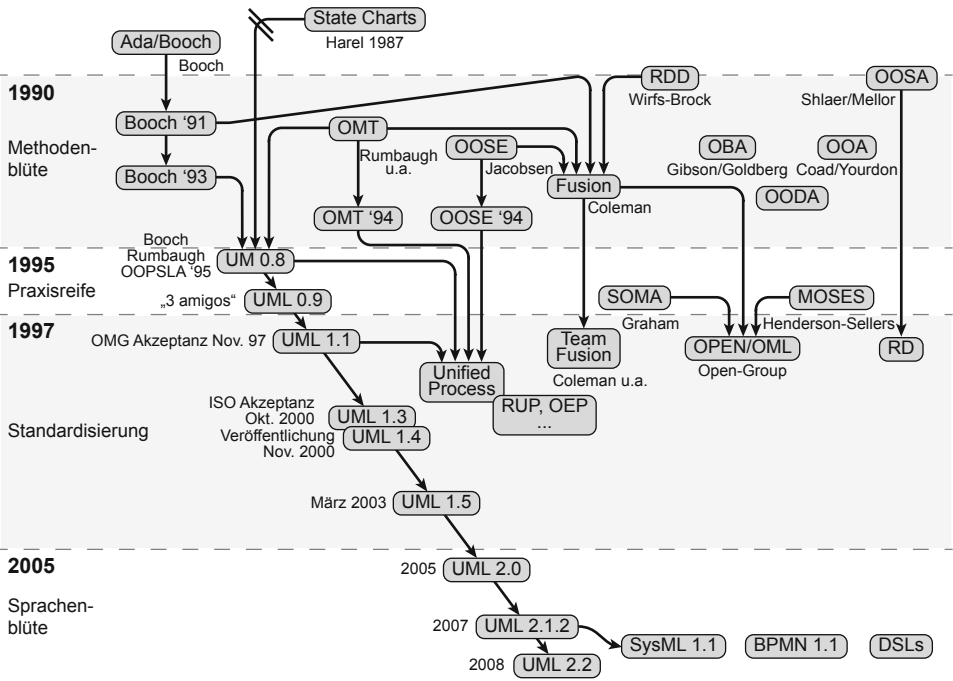


**Abb. 2.23** Die Softwareentwicklung im Kontext des MVPE-Modells: Dokumenten- und Produktmodell (in Anlehnung an [9, 42])

Bei der Weiterentwicklung der Objektorientierung (Abb. 2.24) entwickelten sich einerseits die Sprache UML (Unified Modelling Language) zur Softwareentwicklung als auch Prozessmodelle wie der Unified Software Development Process (USDP) [52]. UML enthält verschiedene Daten-, Zustands und Prozessdiagramme. Einen Überblick über verschiedene UML Diagramme zeigt Abb. 2.25. Aus dem USDP entwickelte die Firma IBM den *Rational Unified Process* (RUP). Dieser Prozess basiert auf dem USDP und ist ebenfalls ein für die Nutzung der Modellierungssprache UML beschriebenes Vorgehensmodell/ Phasenmodell für die Anwendungsentwicklung. Der Rational Unified Process ist als Software Engineering Vorgehensmodell selbst in der UML beschrieben. Das RUP-Modell ist als Sammlung von Best Practices im IBM Rational Method Composer umgesetzt [36].

Der Arbeitsablauf im RUP-Vorgehensmodell ist aus dem evolutionären/inkrementellen Vorgehen abgeleitet. Das Modell ist gekennzeichnet durch

- die iterativ durchgeführte Softwareentwicklung inklusive der Modellierung der Geschäftsprozesse,
- ein integriertes Anforderungsmanagement,
- auf Komponenten basierte Informatik-Architekturen,



**Abb. 2.24** Geschichtliche Entwicklung der Objektorientierung (nach [64])

Zu modellierendes Element <i>Verwendete Diagrammart aus UML</i>	Beschreibung
<b>Funktionale Anforderungen</b> <i>Anwendungsfalldiagramm</i>	Die Anforderungen der Benutzer werden anhand von sinnvollen Einheiten dargestellt.
<b>Modell der Domäne</b> <i>Klassendiagramm</i>	Die statische Struktur der Problemdomäne wird mit Hilfe von Klassen gezeigt.
<b>Objektorientiertes Modell des Systems</b> <i>Klassendiagramm mit Stereotypen für Analysemodell</i>	Die statische Struktur des Systems wird mit Hilfe von Klassen gezeigt.
<b>Objektzusammenarbeit</b> <i>Sequenzdiagramm</i> <i>Kollaborationsdiagramm</i>	Zeigt die dynamische Zusammenarbeit von Objekten.
<b>Objektzustände</b> <i>Zustandsdiagramm</i>	Zeigt die Zustände eines Objektes oder eines Moduls.
<b>Systemarchitektur</b> <i>Pakete, Komponentendiagramm</i> <i>Auslieferungsdiagramm</i>	Zeigt die Elemente der Systemarchitektur in verschiedenen Sichten.
<b>Abläufe</b> <i>Aktivitätsdiagramm</i>	Zeigt einzelne Aktivitäten und den zeitlichen Zusammenhang zwischen den Aktivitäten.

**Abb. 2.25** UML Klassendiagramme



- eine visuelle Software-Modellierung,
- eine verifizierbare Software-Qualität und
- ein kontrolliertes Change-Management.

Der Ablauf eines Projektes teilt sich in vier zeitlich gestaffelte Phasen auf, die in mehreren Arbeitsschritten behandelt werden können [64]:

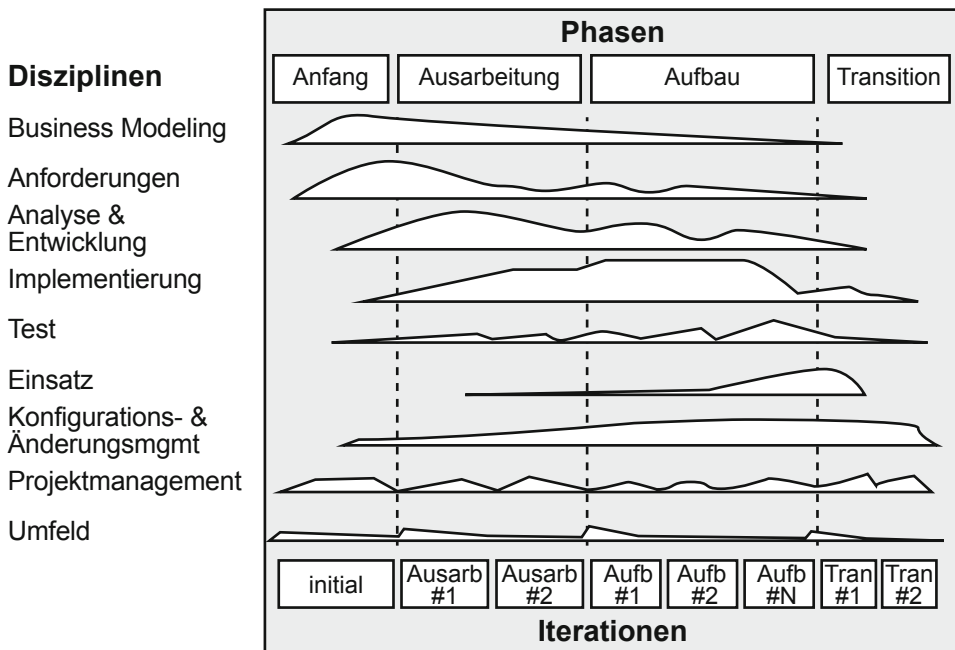
- Phase *Inception* = Initiierung eines Workflow, zum Beispiel Abgrenzung System (zum Beispiel über Kontextdiagramm), Analyse Prozesse, Analyse Anforderungen, Konzeption, Planung Vorgehen, Definition der Qualitätssicherung.
- Phase *Elaboration* = Entwicklung im Workflow, zum Beispiel Modellierung Prozesse, Spezifikation Anforderungen, Fachkonzept, Grobkonzept, Feinkonzept, Entwurf System-Architektur, Entwurf Test-Konzept, Qualitätssicherung.
- Phase *Construction* = Ausarbeitung im Workflow, zum Beispiel Ableitung Anforderungen, Entwurf Software-Architektur, Konstruktion, Realisierung, Implementierung, Test, Abnahme.
- Phase *Transition* = Software-Paketierung, Software-Verteilung (rollout), Inbetriebnahme

Die Arbeitsschritte (Iterationen) des RUP Vorgehensmodells für jede Phase zeigt Abb. 2.26.

Andere objektorientierte Prozessmodelle, die sich zeitlich parallel entwickelt haben, sind beispielsweise Catalysis [14] und Fusion [10]. Das bereits vorgestellte V-Modell wurde ab 1997 ebenfalls für die objektorientierte Entwicklung überarbeitet [13] und das Prozessumfeld gestärkt. Die Softwareentwicklung wird als Folge von Aktivitäten definiert, die in Abb. 2.27 dargestellt sind [[62] in [53]].

In diesem überarbeiteten V-Modell werden innerhalb der Aktivitäten objektorientierte Methoden angewendet. Die Methodenzuordnung stellt für jede Aktivität eine Liste von möglichen Methoden zur Auswahl, die der Softwareentwickler auswählen kann. Zu den Standardmethoden gehören [62]:

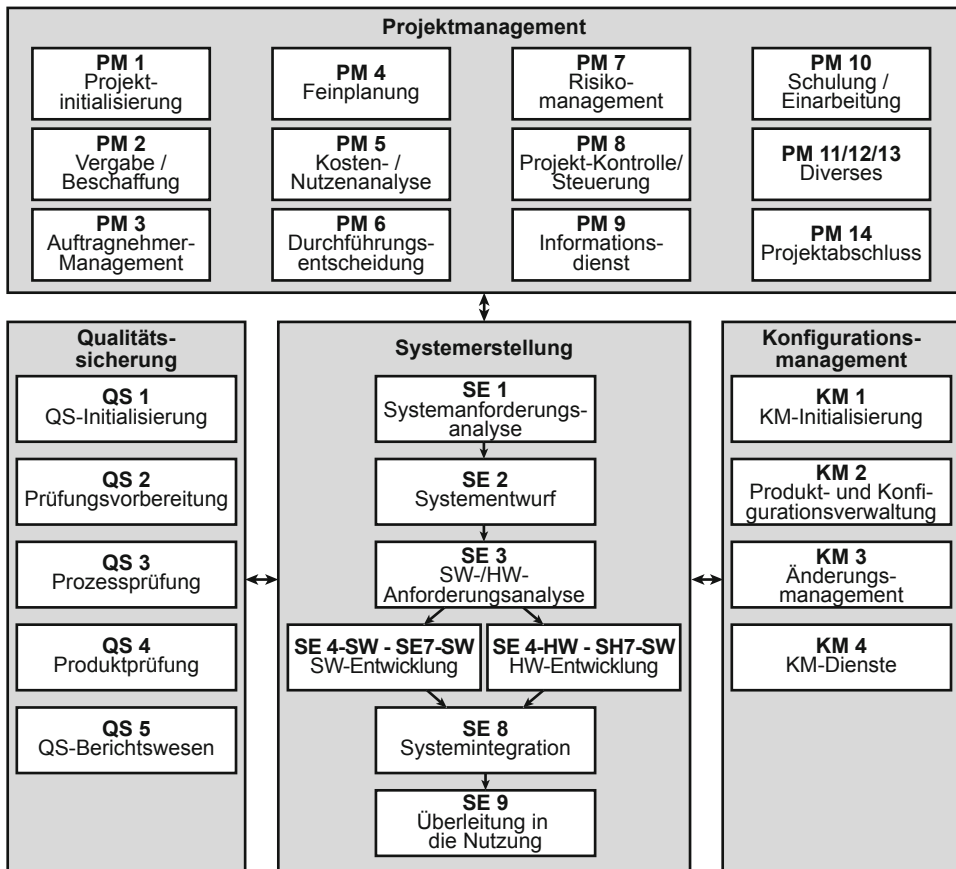
- Usecase Modellierung,
- Klassen und Objekt Modellierung,
- Class-Responsibility-Collaboration Methode,
- Zustandsmodellierung,
- Interaktionsmodellierung,
- Subsystemmodellierung sowie
- Prozess- und Moduldiagramme.



**Abb. 2.26** Phasen und Arbeitsschritte des RUP Vorgehensmodell (nach [36])

**Agile Softwareentwicklung** [8] ist der Sammelbegriff für den Einsatz von Agilität und Flexibilität in der Softwareentwicklung. Je nach Kontext bezieht sich der Begriff auf Teilbereiche der Softwareentwicklung – wie im Fall von Agile Modeling – oder auf den gesamten Softwareentwicklungsprozess. Agile Softwareentwicklung versucht mit geringem bürokratischen Aufwand, wenigen Regeln und meist einem iterativen Vorgehen auszukommen. Das Ziel agiler Softwareentwicklung ist es, den Softwareentwicklungsprozess flexibler und schlanker zu machen, als das bei den klassischen Vorgehensmodellen der Fall ist. Man möchte sich mehr auf die zu erreichenden Ziele fokussieren und auf technische und soziale Probleme bei der Softwareentwicklung eingehen. Die agilen Methoden eignen sich daher besonders gut, um auf geänderte Anforderungen zu reagieren, da die Entwicklungszyklen in der Regel kurz ausgelegt sind. Die agile Softwareentwicklung sieht sich als Gegenbewegung zu den oft als schwergewichtig und bürokratisch angesehenen traditionellen Softwareentwicklungsprozessen wie dem Rational Unified Process oder dem V-Modell [63].

Die Werte agiler Softwareentwicklung bilden das Fundament. Im Februar 2001 haben 17 Erstunterzeichner diese Werte als *Agiles Manifest* (englisch *Manifesto for Agile Software Development* oder kurz *Agile Manifesto*) formuliert [agilemanifesto.org/iso/de/, [1]]:



**Abb. 2.27** V-Modell mit Submodellen und Aktivitäten (nach [13])

#### Definition „agile Softwareentwicklung“:

„Wir zeigen bessere Wege auf, Software zu entwickeln, indem wir es selber tun und anderen dabei helfen, es zu tun. Durch unsere Arbeit sind wir zu folgender Erkenntnis gekommen:

- *Menschen und Interaktionen sind wichtiger als Prozesse und Werkzeuge.*
- *Funktionierende Software ist wichtiger als umfassende Dokumentation.*
- *Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlungen.*
- *Eingehen auf Veränderungen ist wichtiger als Festhalten an einem Plan“ [1].*

Inzwischen gibt es eine Vielzahl von agilen Prozessmodellen (Abb. 2.28).

Der Rational Unified Process wird von vielen Vertretern agiler Methoden als nicht-agiler, schwergewichtiger Prozess aufgefasst. Das ist allerdings umstritten [15] beziehungsweise wurde versucht mit dem *Agile Unified Process* eine agile Variante von RUP zu entwickeln

### Spiralmodell

(Barry W. Boehm 1988, Chan&Leung 1996, Kümmel 1999)

- Integriert Prototypansatz

### eXtreme Programming (XP)

(Kent Beck, Cynthia Andres 2008)

- Intuitive nachvollziehbare Erkenntnisse als Basis

### Crystal

(Alistair Cockburn 1998)

- Sammlung an Methoden

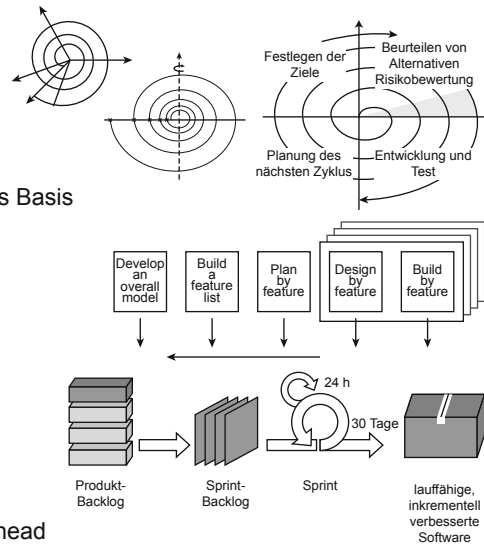
### Feature Driven Development (FDD)

(Jeff De Luca, Peter Coad, Eric Lefebvre 1999)

- Entwicklung anhand Featureplan

### Scrum (engl. Gedränge) (Ken Schwaber 2008)

- Reduzierung des organisatorischen Overhead



**Abb. 2.28** Überblick über agile Softwareentwicklungsmethoden (in Anlehnung an [29])

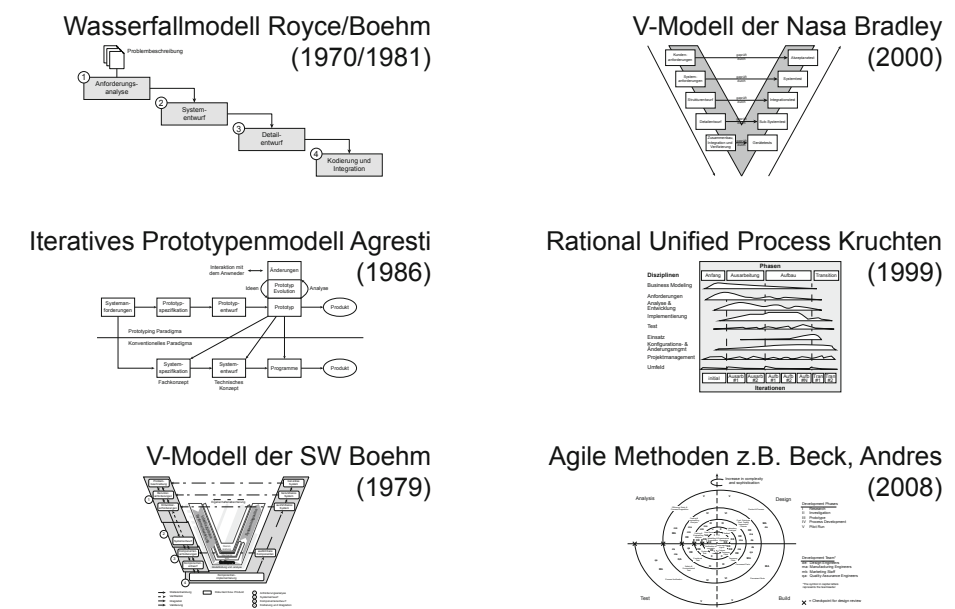
[32]. Eine noch extremere Haltung vertritt Hellige [31]. Seiner Meinung nach wirken alle prozeduralen Dekompositions-orientierten Methoden und Vorgehensmodelle sowohl bei Mechanik und Elektrik/Elektronik als auch bei Software der Kreativität und der Designkomplexität entgegen und zielen vorrangig auf Standardisierung und Prozessrationalisierung ab.

Abbildung 2.29 fasst nochmals die wesentlichen Methoden und Vorgehensmodelle der Softwareentwicklung zusammen.

## 2.4 Disziplin-übergreifende Vorgehensmodelle aus der Mechatronik und dem Systems Engineering

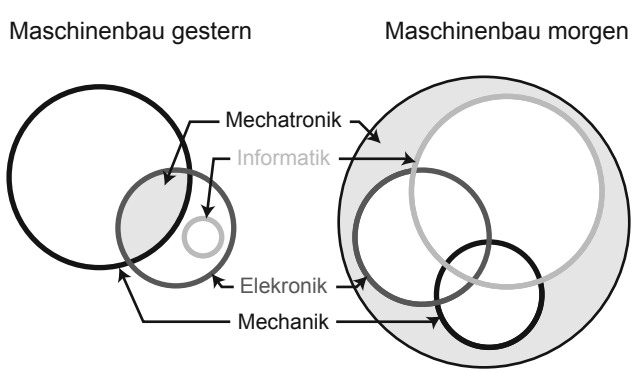
Der Japaner Ko Kikuchi verwendete 1969 zum ersten Mal den Begriff *Mechatronik* [30]. Der Begriff besteht aus Wortbestandteilen der Mechanik und der Elektronik und kennzeichnete zunächst nur die elektrotechnische und elektronische Funktionserweiterung von mechanischen Komponenten und Geräten [27]. Die Software hat ihre Bedeutung in der Mechatronik erst viel später gewonnen (Abb. 2.30). Interessant ist, dass sich aus der anfänglichen Wortkombination ein weltweit anerkanntes, ingenieurwissenschaftliches Arbeitsgebiet entwickelt hat, das den interdisziplinären Systemgedanken in den Mittelpunkt der Produktentwicklung stellt [35].

Mechatronische Methoden und Vorgehensmodelle basieren beispielsweise bei Isermann [34] auf einem Prozessverständnis, auf dem Vorgehen von Pahl Beitz, beispielsweise bei



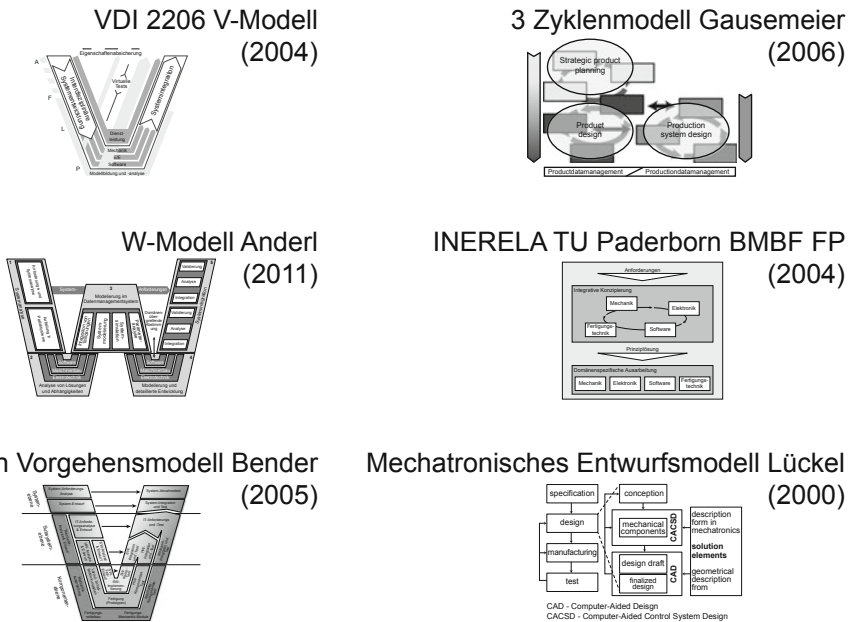
**Abb. 2.29** Überblick über Methoden und Vorgehensmodelle der Softwareentwicklung

**Abb. 2.30** Wandlung des Begriffes Mechatronik (in Anlehnung an [37])



Lückel [43], oder auf verschiedenen Variationen des V-Modells. Das bekannteste ist die VDI Richtlinie 2206, die ein flexibles Vorgehen mit den drei Elementen „Problemlösungszyklus als Mikrozyklus“ [12], „V-Modell als Makrozyklus“ und „Prozessbausteine für wiederkehrende Arbeitsschritte“ [59] beinhaltet. Vom V-Modell abgeleitete und teilweise stärker ausdetaillierte Vorgehensmodelle wurden von Bender [6] und Anderl (↗ W-Modell) [2] vorgestellt.

Um die bei mechatronischen Produkten aufgrund der Integration mechanischer, elektrotechnischer und softwaretechnischer Komponenten erhöhten Anforderungen sowohl in der Produktentstehung als auch in der Fertigungsplanung zu beherrschen, wurde von Gausemeier das 3-Zyklenmodell der Produktentstehung erarbeitet. Das Vorgehensmodell



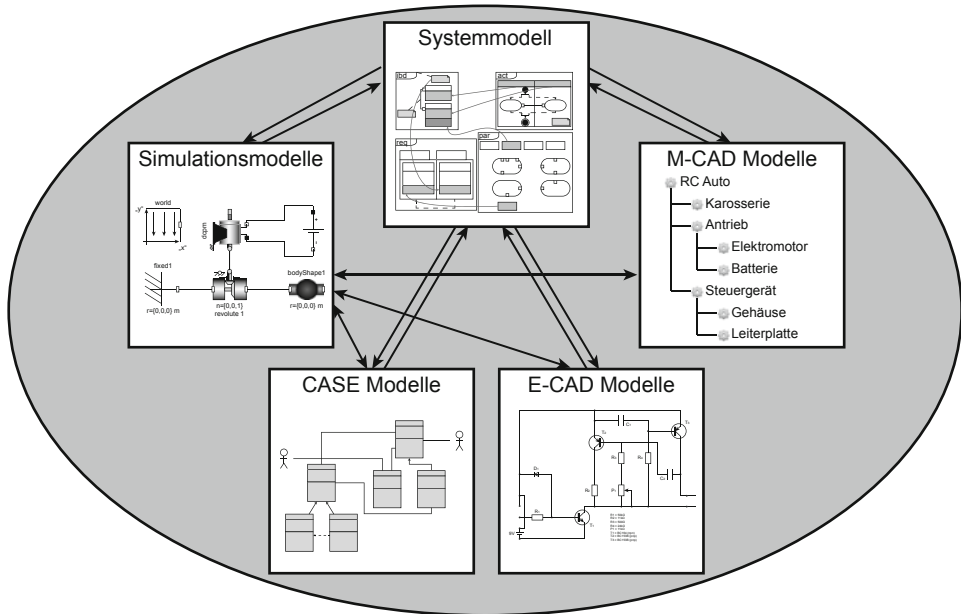
**Abb. 2.31** Überblick über Methoden und Vorgehensmodelle der Mechatronik

umfasst hierbei die Funktionsbereiche strategische Produktplanung, Produktentwicklung und Produktionssystementwicklung, die jeweils zyklisch durchlaufen werden [26].

Im Rahmen des Verbundprojektes „INERELA – Integrative Entwicklung räumlicher elektronischer Bauteile“ wurde eine Vorgehensweise festgelegt, die nicht nur die Produktentwicklung, sondern auch die Entwicklung des Fertigungssystems berücksichtigt. Das Vorgehensmodell lehnt sich an die VDI-Richtlinie 2206 an und berücksichtigt weitere Entwicklungsmodelle wie das Vorgehensmodell nach Pahl/Beitz und das Y-Modell nach Gajski. Das Vorgehensmodell wird hierbei in zehn Phasen unterteilt, die wiederum jeweils mehrere Prozessschritte beinhalten [54]. Einen Auszug aus der Vielzahl der Methoden und Vorgehensmodelle der Entwicklung mechatronischer Produkte zeigt Abb. 2.31.

Parallel wurde seit den 1960er Jahren insbesondere bei der amerikanischen Luft- und Raumfahrt und in großen Militärprojekten *Systems Engineering* (SE) als interdisziplinärer, dokumentengetriebener Ansatz zur Entwicklung und Umsetzung komplexer, technischer Systeme definiert. Dieser Ansatz wurde aus Sicht der Software- und Elektronikindustrie permanent ausgebaut und bietet heute Modellierungs- und Simulationsunterstützung von komplexen, stark vernetzten Systemen an. Nach den Vorgaben der INCOSE<sup>2</sup> ist das Systems Engineering wie folgt definiert:

<sup>2</sup> International Council on Systems Engineering



**Abb. 2.32** Modellbasierte Produktentwicklung (in Anlehnung an [23])

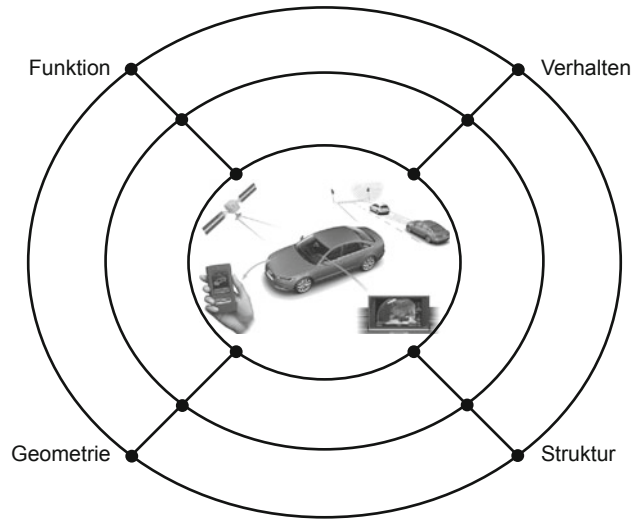
#### Definition „Systems Engineering“:

Systems Engineering ist eine Disziplin, deren Aufgabe die Erstellung und Ausführung eines interdisziplinären Prozesses ist, der garantieren soll, dass Kunden- und Stakeholder-Anforderungen qualitativ hochwertig, zuverlässig, kostengünstig und in vorgegebener Zeit über den gesamten Produktlebenszyklus erfüllt werden können [33].

Während klassische Methoden des Systems Engineerings papier- oder dokumentenbasiert sind, ermöglicht *Model-Based Systems Engineering (MBSE)* als Weiterführung des Systems Engineerings ein modellbasierendes Vorgehensmodell [24]. Es ist ein multidisziplinärer Ansatz und basiert auf entwicklungsphasenspezifischen, digitalen Systemmodellen, die entlang des Produktentwicklungsprozesses integriert werden und erlaubt die Modellierung in verschiedenen Phasen des Produktentwicklungsprozesses. (Abb. 2.32).

Das Problem der Integration einzelner Komponenten während des Entwicklungsprozesses kann durch die Verwendung solcher Modellierungssprachen möglichst früh in Angriff genommen werden, indem die Korrelationen zwischen Systemanforderungen, Funktionen, Verhalten und Struktur definiert werden. Die durchgängige, modellbasierte Entwicklung ist in der virtuellen Produktentwicklung von zentraler Bedeutung und ist somit auch eine

**Abb. 2.33** X-Modell als mögliche Methode für die interdisziplinäre Produktentwicklung



wesentliche Herausforderung an die Optimierung des PEP für mechatronische und insbesondere für cybertronische Produkte beziehungsweise Systeme. Ein aus dem Y-Modell von Gajski und auf dem Funktions-Verhaltens-Struktur Modellierungsansatz von Umeda et.al. [55] und Gero [28] abgeleitetes X-Modell zeigt Abb. 2.33.

Diese Methode korrespondiert mit dem in Kapitel 4 vorgestellten MBSE Vorgehensmodell und könnte potentiell einen gemeinsamen Ansatz einer Entwurfsmethode für alle drei mechatronischen Disziplinen bilden. Durch Wegfall je einer Achse, können die verschiedenen Disziplinen repräsentiert werden:

- Mechanik: Wegfall Verhalten (der Mechaniker beschreibt das Verhalten nicht detailliert im Rahmen der Anforderungsmodellierung, sondern ermittelt es im Rahmen der Simulation)
- Elektronik (evtl. Wegfall der Funktion)
- Software (Wegfall der Geometrie)

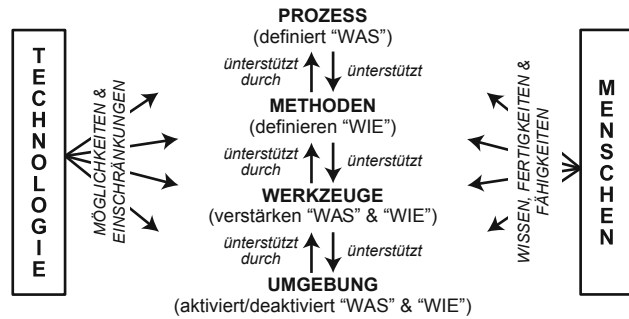
Die dazugehörigen Definitionen sind:

#### Definitionen

Funktion	Die Absicht oder Zweck der Entwicklung eines Produktes oder beliebiger untergeordneter Komponenten und Funktionsträgern
Verhalten	Wie das Konstruktionsobjekt die Funktion erreicht oder umsetzt
Struktur	Die Beschreibung der Konstruktionsobjekte in der jeweiligen Lebenszyklusphase und ihre Beziehungen zueinander in horizontaler sowie



**Abb. 2.34** Begriffsdefinition für Vorgehensmodelle MBSE (in Anlehnung an [20, 46])



in vertikaler Richtung top down und bottom up (Auflösung und Verwendung)

**Geometrie** Die räumliche und topologische zwei- bzw. dreidimensionale Ausdehnung und Positionierung der Konstruktionsobjekte

Im „Survey of MBSE Methodologies“ [20] wird zunächst eine grundsätzliche Definition von Prozess, Methode, Werkzeug und Umgebung vereinbart (Abb. 2.34).

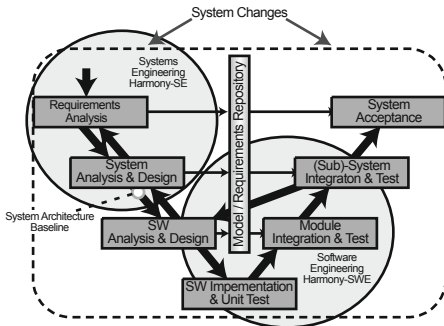
Danach gelten folgende Definitionen [20]:

#### Definitionen

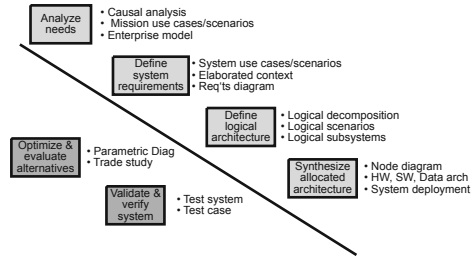
Prozess	Ein Prozess ist eine logische Folge von Tätigkeiten, um ein Ziel zu erreichen. Er definiert, „was“ zu tun ist und nicht „wie“ etwas zu tun ist.
Methode	Eine Methode besteht aus Techniken, um diese Tätigkeiten umzusetzen. Sie definiert das „wie“.
Werkzeug	Ein Werkzeug ist in der Regel eine Softwarelösung, die, auf eine Methode angewandt, deren Umsetzung garantiert und die Effizienz garantiert.
Umgebung	Eine Umgebung besteht aus externen Objekten, individuellen Personen oder Gruppen und Bedingungen beispielsweise sozialer, organisatorischer, funktionaler oder kultureller Art.

Basierend auf diesen Definitionen ist ein Vorgehensmodell (engl.: „methodology“) der Zusammenhang und das Zusammenwirken von Prozess, Methode, Werkzeug und Umgebung. In den letzten Jahren wurden diverse Vorgehensmodelle für MBSE vorgestellt (Abb. 2.35).

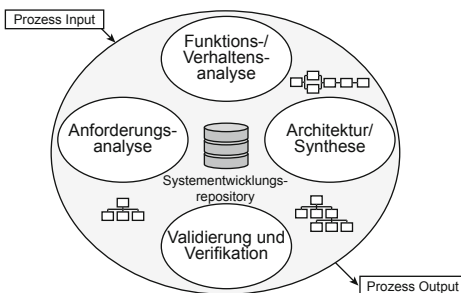
## IBM Telelogic Harmony-SE



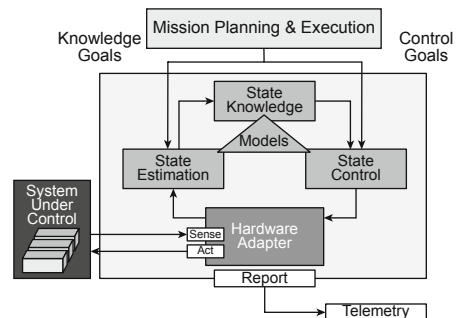
## OOSEM (INCOSE)



## Vitech MBSE



## JPL State Analysis



**Abb. 2.35** Überblick über Methoden und Vorgehensmodelle des Systems Engineerings (nach [20])

### Übungsaufgaben

- Was sind die Hauptphasen des PEP im Maschinenbau?
- Erklären Sie den Inhalt der VDI Norm 2221.
- Erklären sie die Funktionsdefinition nach Pahl & Beitz.
- Erklären Sie die Begriffe physikalischer Effekt und Wirkprinzip.
- Wie sieht das Verhältnis von Kostenverantwortung und -verursachung aus?
- Welche Methoden und Vorgehensmodelle im Maschinenbau kennen Sie?
- Was sind die Grundunterschiede der Vorgehensmodelle des Maschinenbaus und der Elektrotechnik/Elektronik?
- Welche Anwendungsgebiete in der Elektrotechnik/Elektronik kennen Sie?
- Welche Methoden und Vorgehensmodelle in der Elektrotechnik/Elektronik kennen Sie?
- Was ist das Besondere des Gajski Diagramms?
- Was sind Entwurfsebenen elektronischer Systeme?
- Welche Methoden und Vorgehensmodelle der Softwareentwicklung kennen Sie?

- Welche Phasen und Arbeitsschritte beinhaltet das RUP Modell?
- Was versteht man unter agiler Softwareentwicklung?
- Erklären Sie den Begriff Mechatronik.
- Welche Methoden und Vorgehensmodelle der Mechatronik kennen Sie?
- Erläutern Sie den Begriff modellbasierte Produktentwicklung nach Friedenthal.
- Was sind Begriffsdefinitionen für Vorgehensmodelle?
- Erklären Sie die Begriffe Funktion, Verhalten, Struktur und Geometrie.
- Erklären Sie die Begriffe Prozess, Methode, Werkzeug und Umgebung?
- Welche MBSE Vorgehensmodelle kennen Sie?

---

## Literatur

1. 10 Jahre Agiles Manifest: OBJEKTSpektrum. Nr. 2 Schwerpunkttheft (2011)
2. Anderl, R., Nattermann, R., Rollmann, T. (Hrsg.): Das W-Modell – Systems Engineering in der Entwicklung aktiver Systeme. <http://www.plmportal.org/forschung-details/items/das-w-modell-systems-engineering-in-der-entwicklung-aktiver-systeme.html>. Zugegriffen: 12. April 2013.
3. Andreasen M.M.: Machine Design Methods Based on a Systematic Approach. Dissertation. Lund University, Sweden (1980)
4. Andreasen M.M., Hein L.: Integrated Product Development. IFS, Bedford (1987)
5. Balzert, H.: Lehrbuch der Software-Technik, Lehrbücher der Informatik, 2. Aufl., Bd. 1. Spektrum Akademischer, Heidelberg (2001)
6. Bender, K.: Embedded Systems – qualitätsorientierte Entwicklung. Springer, Heidelberg (2005)
7. Blanchard, Benjamin S.: Systems Engineering and Analysis: Pearson New International Edition. 5. Aufl. Harlow: Pearson Education Limited. (2013)
8. Bleek, W., Wolf, H.: Agile Softwareentwicklung. Werte, Konzepte und Methoden, 1. Aufl. it-agile. dpunkt, Heidelberg (2008)
9. Boehm, B.: Guidelines for Verifying and Validating Software Requirements and Design Specifications. In: Samet, P.A. (Hrsg.) Euro IFIP 79. North-Holland Publishing Company, Amsterdam (1979)
10. Coleman, D.: Object Oriented Development: The Fusion Method. Prentice Hall, Upper Saddle River, New Jersey (1994)
11. Cross, N.: Engineering Design Methods. Strategies for Product Design, 2. Aufl. Wiley, Chichester (1994)
12. Daenzer, W.F., Huber, F. (Hrsg.): Systems Engineering – Methoden und Praxis. Verlag Industrielle Organisation, Zürich (1994)
13. Dorschel, W., Heuser, W., Midderhoff, R.: Inkrementelle und objektorientierte Vorgehensweise mit dem V-Modell. Oldenbourg, München (1998)
14. D'Souza, D.F., Willa, A.C.: Objects, Components and Frameworks with UML. Addison-Wesley, Boston (1998)
15. Eckstein, J.: Agile Softwareentwicklung im Großen. Ein Eintauchen in die Untiefen erfolgreicher Projekte. Deutsche Bearbeitung von Nicolai Josuttis. dpunkt, Heidelberg (2004)
16. Eder, W.E., Hosnedl, S.: Design Engineering. A Manual for Enhanced Creativity. CRC Press, Boca Raton (2008)

17. Ehrlenspiel, K.: Integrierte Produktentwicklung. Methoden für Prozeßorganisation Produkterstellung und Konstruktion. Hanser, München (1995)
18. Ehrlenspiel, K.: Integrierte Produktentwicklung. Denkabläufe, Methodeneinsatz, Zusammenarbeit, 3. Aufl. Hanser, München (2007)
19. Eigner, M., Gerhardt, F., Gilz, T., Mogo Nem, F.: Informationstechnologie für Ingenieure. Springer Vieweg, Berlin Heidelberg (2012)
20. Estefan, J.: Survey of Candidate Model-Based Systems Engineering (MBSE) Methodologies, rev. B. Pasadena, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TD-2007-003-02, 23 May 2008
21. Feldhusen, J.: Konstruktionslehre I und II, Vorlesungsunterlagen. RWTH Aachen University, Aachen (2012)
22. French, M.J.: Conceptual Design For Engineers, 3. Aufl. Springer, Heidelberg (1999)
23. Friedenthal, S., Greigo, R., Sampson, M.: INCOSE MBSE Roadmap, in „INCOSE Model Based Systems Engineering (MBSE) Workshop Outbrief“, S. 6. Paper presented at INCOSE International Workshop 2008, Albuquerque, NM, 26. Jan 2008
24. Friedenthal, S., Steiner, R., Moore, A.: A Practical Guide to SysML – The Systems Modeling Language. Morgan Kaufmann, San Francisco (2009)
25. Gajski, D.D.: Construction of a large scale multiprocessor. Urbana, Ill: Cedar Project, Laboratory for Advanced Supercomputers, Dept. of Computer Science, University of Illinois at Urbana-Champaign (Report/Department of Computer Science, University of Illinois at Urbana-Champaign, no. UIUCDCS-R-83-1123), 1983
26. Gausemeier, J., Ebbesmeyer, P., Kallmeyer, F.: Produktinnovation. Strategische Planung und Entwicklung der Produkte von Morgen. Hanser, München (2001)
27. Gausemeier, J., Redenius, A.: Entwicklung mechatronischer Systeme. In: Schäppi, B., Andreasen, M.M., Kirchgeorg, M., Radermacher, F.J. (Hrsg.): Handbuch Produktentwicklung. Verlag Hanser, Wien München (2005)
28. Gero, J., Tham, K., Lee, H. Behaviour: A Link Between Function and Structure in Design: Intelligent computer Aided Design. Elsevier, Amsterdam (1992)
29. Gilz, T.: PLM-Integrated Interdisciplinary System Models in the Conceptual Design Phase Based on Model-Based Systems Engineering. Dissertation. Technische Universität Kaiserslautern. Schriftenreihe VPE. Band 13. Kaiserslautern. 2014
30. Harashima, F.: Mechatronics – „What Is It, Why, and How?“ An Editorial. IEEE/ASME. Trans. Mechatron. **1**, 1–4 (1996)
31. Hellige, H.-D.: Wissenschaft vs. Design: Konstruktionslehren für den Maschinenbau, den Computer und die Software im historischen Diskursvergleich. artec-paper Nr. 153, März 2008
32. Hruschka, P., Rupp, C., Starke, G.: Agility kompakt. Tipps für erfolgreiche Systementwicklung. Spektrum, Akademischer Verlag, Heidelberg (2003)
33. INCOSE – A Consensus of the INCOSE Fellows: <http://www.incose.org/practice/fellowsconsensus.aspx>, zuletzt geprüft. Zugriffen: 19. März 2013
34. Isermann, R.: Mechatronische Systeme. Grundlagen. Springer, Heidelberg (1999)
35. Janschek, K.: Systementwurf mechatronischer Systeme. Methoden – Modelle – Konzepte. Springer, Heidelberg (2010)
36. Kruchten, P.: The rational Unified Process (Addison-Wesley object technology series). Addison-Wesley, Reading (1999)
37. Kühnl, C.: Software gibt den Takt vor. Mechatron. Eng. **2**, 24–25 (2010)
38. Kümmel, M.A.: Integration von Methoden und Werkzeugen zur Entwicklung von mechatronischen Systemen. Dissertation, Universität-Gesamthochschule Paderborn (1999)
39. Lehmann, G., Wunder, B., Selz, M.: Schaltungsdesign mit VHDL. Synthese, Simulation und Dokumentation digitaler Schaltungen. Franzis, Haar (1994)
40. Lienig, J.: Layoutsynthese elektronischer Schaltungen – Grundlegende Algorithmen für die Entwurfsautomatisierung, 1. Aufl. Springer, Heidelberg (2006)

41. Liggesmeyer, P.: Grundlagen Software Engineering, Vorlesungsunterlagen, AG Software Engineering, TU Kaiserslautern (2013)
42. Lind, J.: Iterative Software Engineering for Multiagent Systems. The MASSIVE Method. Springer, Heidelberg (2001)
43. Lückel, J.: Entwicklungsumgebungen Mechatronik. Methoden und Werkzeuge zur Entwicklung mechatronischer Systeme. HNI (HNI-Verlagsschriftenreihe, 80), Paderborn (2000)
44. Lüdecke, A.: Simulationsgestützte Verfahren für den Top-Down-Entwurf heterogener Systeme. Dissertation, Universität Duisburg-Essen (2003)
45. Malmqvist, J., Svensson, D. (Hrsg.): A Design Theory Based Approach Towards Including QFD Data In Product Models. Proceedings of the 1999 ASME Design Engineering Technical Conferences. Las Vegas, Nevada, USA (1999)
46. Martin, J.N.: Systems Engineering Guidebook: A Process for Developing Systems and Products. CRC Press, Boca Raton (1996)
47. Pahl, G., Beitz, W.: Konstruktionslehre Methoden und Anwendung, 4. Aufl. Springer, Heidelberg (1997)
48. Pahl, G., Beitz, W.: Konstruktionslehre Grundlagen erfolgreicher Produktentwicklung; Methoden und Anwendung, 7. Aufl. Springer, Heidelberg (2007)
49. Ponn, J., Lindemann, U. (Hrsg.): Konzeptentwicklung und Gestaltung technischer Produkte. Springer, Heidelberg (2011)
50. Pugh, S.: Total Design Integrated Methods for Successful Product Engineering. Prentice Hall, Upper Saddle River, New Jersey (1990)
51. Rauscher, R.: Entwurfsmethodik hochintegrierter anwendungsspezifischer digitaler Systeme, 1. Aufl. Pro Universitate Verlag, Sinzheim (1996)
52. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual (The Addison-Wesley object technology series). Addison-Wesley, Reading (1999)
53. Schäppi, B., Andreasen, M.M., Kirchgeorg, M., Radermacher, F.J. (Hrsg.): Handbuch Produktentwicklung. Hanser, Wien München (2005)
54. Stephan, N.: Vorgehensmodell zur Unterstützung der interdisziplinären und föderierten Zusammenarbeit in der frühen Phase der Produktentstehung am Beispiel der Nutzfahrzeugindustrie. Dissertation, FB MV TU Kaiserslautern (2013)
55. Umeda, Y., Tanaka, H., Tomiyama, T. and Yoshikawa H.: Function, Behaviour and Structure. Proceedings of the Fifth International Conference, Boston (1990)
56. VDI-Richtlinie 2235: Wirtschaftliche Entscheidungen beim Konstruieren. Düsseldorf (1987)
57. VDI-Richtlinie 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. Düsseldorf (1993)
58. VDI-Richtlinie 2422: Entwicklungsmethodik für Geräte mit Steuerung durch Mikroelektronik. Düsseldorf (1994)
59. VDI-Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme. Düsseldorf (2004)
60. Walker, R.: Applied Qualitative Research. Gower, Aldershot (1985)
61. Wehn, N.: System Modelling HW/SW Co-Design Optimization, Vorlesungsunterlagen TU Kaiserslautern, Lehrstuhl Entwurf Mikro- elektronischer Systeme (2013)
62. Weisbecker, A.: Softwareentwicklung. In: Schäppi, B., Andreasen, M.M., Kirchgeorg, M., Radermacher, F.J. (Hrsg.) Handbuch Produktentwicklung. Hanser, Wien München (2005)
63. Wikipedia Agile Softwareentwicklung: [http://de.wikipedia.org/wiki/Agile\\_Softwareentwicklung](http://de.wikipedia.org/wiki/Agile_Softwareentwicklung). Zugriffen: 23. März 2014
64. Wikipedia Rational Unified Process: [http://de.wikipedia.org/wiki/Rational\\_Unified\\_Process](http://de.wikipedia.org/wiki/Rational_Unified_Process). Zugriffen: 23. März 2014
65. Wikipedia SystemC: <http://de.wikipedia.org/wiki/SystemC>. Zugriffen: 23. März 2014
66. Wynn, D., Clarkson, J.: Models of Design, Chapter 1, University of Cambridge, 2004

Modellbasierte virtuelle Produktentwicklung

Eigner, M.; Roubanov, D.; Eigner, M. (Hrsg.)

2014, X, 401 S. 336 Abb., Hardcover

ISBN: 978-3-662-43815-2